aws

用户指南

Amazon Linux 2023

Amazon Linux 2023: 用户指南

Table of Contents

什么是 Amazon Linux 2023？

Amazon Linux 2023（AL2023）是 Amazon Web Services (AWS) 提供的下一代 Amazon Linux。借助 AL2023，您可以在安全、稳定和高性能的运行时环境中开发和运行云及企业应用程序。您还可以获得一个提供长期支持并能访问 Linux 最新创新的应用程序环境。AL2023 是免费提供的。

AL2023 是 Amazon Linux 2（AL2）的后继版本。有关 AL2023 和 AL2 之间差异的信息，请参阅 比较 AL2 和 AL2 023 和 AL2023 中的程序包变更。

主题

发布频率

Amazon Linux 2023（AL2023）于 2023 年 3 月发布，并将支持到 2029 年 6 月 30 日。支持分为两个阶段：

- 标准支持：在此阶段，该发布版本会收到季度的次要版本更新。标准支持阶段于 2027 年 6 月 30 日结束。
- 维护：在此阶段，该发布版本仅接收安全更新和关键错误修复。这些更新在可用时会立即发布。维护阶段于 2029 年 6 月 30 日结束。

主要和次要发布

伴随每个 Amazon Linux 发布（主要版本、次要版本或安全发布），我们都会发布一个新的 Linux Amazon 机器映像 (AMI)。

- 主要版本发布 — 包括整个堆栈的安全和性能的新功能和改进。改进可能包括对内核、工具链、Glib C、OpenSSL 以及任何其他系统库和实用程序的重大更改。Amazon Linux 的主要发布部分基于上游 Fedora Linux 发行版的当前版本。AWS 可能会添加或替换来自其他非 Fedora 上游的特定软件包。
- 次要版本发布 — 季度更新，其中包括安全更新、错误修复以及新功能和软件包。每个次要版本都是更新的一个累积列表，其中除了包括新功能和软件包，还包括安全和错误修复。这些发布可能包括最新的语言运行时，例如 PHP。还可能包括其他流行的软件包，例如 Ansible 和 Docker。

使用新版本

更新是通过新的 Amazon 机器映像 (AMI) 版本与相应的新存储库的组合提供的。默认情况下，新的 AMI 与它所指向的存储库耦合在一起。但是，随着时间的推移，您可以将正在运行的 Amazon EC2 实例指向更新的存储库版本，以便在正在运行的实例上应用更新。您也可以通过启动最新 AMI 的新实例来更新。

长期支持政策

Amazon Linux 会在一个主要版本中为您基于 Amazon Linux 构建的应用程序提供所有软件包更新并保持兼容性。像 glibc 库、OpenSSL、OpenSSH 和 DNF 软件包管理器之类的核心软件包将获得主要 AL2023 版本的生命周期期限的支持。不属于核心软件包的软件包将基于其特定的上游来源获得支持。您可以通过运行以下命令查看各个软件包的具体支持状态和日期。

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

您可以通过运行以下命令了解当前已安装的所有软件包的信息。

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

核心软件包的完整列表在预览期间最终确定。如果您想了解更多的核心软件包，请告诉我们。我们会在收集反馈时作出评估。如有关于 AL2023 的反馈，可以通过您的指定 AWS 代表提供，也可以通过在 GitHub 上的 amazon-linux-2023 存储库中提交问题来提供。

命名和版本控制

AL2023 在为期两年的标准支持期内，会每三个月提供一个次要版本。每个版本都用从 0 到 N 的一个增量标识。0 指该迭代的最早的主要版本。所有版本都称为 Amazon Linux 2023。当下一版本的 Amazon Linux 发布时，AL2023 将进入延伸支持期，并将收到有关安全更新和关键错误修复的更新。

例如，AL2023 的次要版本具有以下格式：

- 2023.**0**.20230301
- 2023.**1**.20230601
- 2023.**2**.20230901

相应的 AL2023 AMI 具有以下格式：

- 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

在一个特定的次要版本中，常规 AMI 发布使用 AMI 发布日期的时间戳。

- 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

要识别 AL2 或 AL2023 实例，建议您首先读取来自 /etc/system-release-cpe 的通用平台枚举（CPE）字符串。然后，将该字符串拆分为多个字段。最后，读取平台和版本值。

AL2023 还为平台识别引入了新的文件：

- /etc/amazon-linux-release 符号链接到 /etc/system-release
- /etc/amazon-linux-release-cpe 符号链接到 /etc/system-release-cpe

这两个文件表明实例是 Amazon Linux。除非您想知道特定的平台和版本值，否则无需读取文件或将字符串拆分为多个字段。

性能和操作优化

Amazon Linux 6.1 内核

- AL2023 使用适用于弹性网络适配器（ENA）和弹性结构适配器（EFA）设备的最新驱动程序。AL2023 专注于 Amazon EC2 基础设施中硬件的性能和功能反向移植。
- 内核实时修补适用于 x86_64 和 aarch64 实例类型。这减少了频繁重启的需求。
- 所有内核构建和运行时配置都包含许多与 AL2 相同的性能和操作优化。

基本工具链选择和默认构建标志

- AL2023 程序包默认启用编译器优化 (-O2) 进行构建
- 对于 x86-64 系统 (-march=x86-64-v2)，构建 AL2023 软件包需要 x86-64v2，对于 aarch64 (-march=armv8.2-a+crypto -mtune=neoverse-n1)，需要 Graviton 2 或更高版本。
- 构建 AL2023 软件包时启用自动矢量化功能 (-ftree-vectorize)。
- 构建 AL2023 软件包时启用链接时间优化 (LTO)。
- AL2023 使用 Rust、Clang/LLVM 和 Go 的更新版本。

软件包选择和版本

- 主要系统组件的精选反向移植包括针对在 Amazon EC2 基础设施上运行的实例（尤其是 Graviton 实例）的多项性能改进。
- AL2023 集成了多项 AWS 服务和功能。其中包括 AWS CLI、SSM Agent、Amazon Kinesis Agent 和 CloudFormation。
- AL2023 使用 Amazon Corretto 作为 Java 开发工具包 (JDK)。
- 当上游项目发布新版本时，AL2023 为新版本提供数据库引擎和编程语言运行时更新。当发布新版本时，将添加有关新版本的编程语言运行时。

云环境中的部署

- 基本 AL2023 AMI 和容器映像会经常更新，以支持修补实例更换。
- 内核更新包含在 AL2023 AMI 更新中。这意味着您无需使用像 yum update 和 reboot 这样的命令就能更新内核。
- 除了标准 AL2023 AMI，AMI 最低版本 和容器映像也可用。如果选择 AMI 最低版本，则使用运行服务所需的最少量软件包来运行环境。

- 默认情况下，AL2023 AMI 和容器被锁定到特定版本的软件包存储库。当它们启动时，不会自动更新。这意味着您始终掌控着何时提取任何软件包更新。在投入生产之前，您总是可以先在 beta/gamma 环境中测试一下。如果出现问题，可以使用预先验证的回滚路径。

与 Fedora 的关系

AL2023 保持自己独立于 Fedora 的发布和支持生命周期。AL2023 提供开源软件、各种软件包和频繁发布的更新版本。这保留了熟悉的基于 RPM 的操作系统。

AL2023 的正式发布 (GA) 版本并不直接与任何特定的 Fedora 版本步调一致。AL2023 GA 版本包括来自 Fedora 34、35 和 36 的组件。有些组件与 Fedora 中的组件相同，有些则经过修改。其他组件更接近于 CentOS Stream 9 中的组件或是独立开发的。Amazon Linux 内核源自 kernel.org 上长期支持的且独立于 Fedora 选择的选项。

自定义 cloud-init

cloud-init 软件包是一个开源应用程序，可在云计算环境中引导 Linux 映像。有关更多信息，请参阅 cloud-init 文档。

AL2023 包含自定义版本的 cloud-init。使用 cloud-init，您可以指定在引导时您的实例会发生什么情况。

当您启动实例时，可以使用用户数据字段将操作传递给 cloud-init。这意味着，您可以在许多使用案例中使用通用的 Amazon 机器映像 (AMI)，并在启动实例时动态配置它们。AL2023 还使用 cloud-init 来配置 `ec2-user` 账户。

AL2023 使用 `/etc/cloud/cloud.cfg.d` 和 `/etc/cloud/cloud.cfg` 中的 cloud-init 操作。您可以在 `/etc/cloud/cloud.cfg.d` 目录中创建自己的 cloud-init 操作文件。Cloud-init 按字典顺序读取此目录中的所有文件。时间较晚的文件覆盖时间较早的文件中的值。当 cloud-init 启动一个实例时，cloud-init 软件包会执行以下配置任务：

- 设置默认区域
- 设置主机名
- 解析和处理用户数据
- 生成主机私有 SSH 密钥
- 将用户的公有 SSH 密钥添加到 `.ssh/authorized_keys` 以便于登录和管理

- 准备存储库以进行软件包管理
- 处理用户数据中定义的软件包操作
- 运行用户数据中的用户脚本
- 装载实例存储卷（如果适用）
  - 默认情况下，如果 ephemeral0 实例存储卷存在且包含一个有效文件系统，则说明实例存储卷已装载在 /media/ephemeral0。否则，说明未装载。
  - 默认情况下，对于 m1.small 和 c1.medium 实例类型，将装载与实例关联的所有交换卷。
  - 您可以使用以下 cloud-init 指令覆盖默认实例存储卷装载：

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

    有关如何加强对装载的控制，请参阅 cloud-init 文档中的装载。
  - 当一个实例启动时，支持 TRIM 的实例存储卷不会被格式化。要装载实例存储卷，您必须对实例存储卷进行分区和格式化。

    有关更多信息，请参阅《Amazon EC2 用户指南》中的实例存储卷 TRIM 支持。
  - 当您启动实例时，可以使用 disk_setup 模块对实例存储卷进行分区和格式化。

    有关更多信息，请参阅 cloud-init 文档中的磁盘设置。

有关通过 SELinux 使用 cloud-init 的信息，请参阅使用 cloud-init 启用 enforcing 模式。

有关 cloud-init 用户数据格式的信息，请参阅 cloud-init 文档中的用户数据格式。

安全更新和功能

AL2023 提供许多安全更新和解决方案。

主题

- SSH 服务器默认
- OpenSSL 3 的主要功能

管理更新

使用 DNF 和存储库版本应用安全更新。有关更多信息，请参阅 在中管理软件包和操作系统更新 AL2023。

云中的安全性

安全性是 AWS 和您的共同责任。责任共担模型将此描述为云的安全性和云中的安全性。有关更多信息，请参阅 Amazon Linux 2023 中的安全性与合规性。

SELinux 模式

在 AL2023 中，默认启用 SELinux 并设置为许可模式。在许可模式下，记录但不强制执行权限拒绝。

SELinux 策略定义了用户、进程、程序、文件和设备的权限。使用 SELinux，您可以选择两种策略之一。这些策略有针对性或为多级安全 (MLS)。

有关 SELinux 模式和策略的更多信息，请参阅为 AL2023 设置 SELinux 模式和 SELinux 项目维基。

合规性计划

独立审计员将评测 AL2023 以及许多 AWS 合规性计划的安全性和合规性。

SSH 服务器默认

AL2023 包括 OpenSSH 8.7。OpenSSH 8.7 默认禁用 `ssh-rsa` 密钥交换算法。有关更多信息，请参阅 默认 SSH 服务器配置。

OpenSSL 3 的主要功能

- 证书管理协议（CMP，RFC 4210）同时包括 CRMF (RFC 4211) 和 HTTP 传输 (RFC 6712)。
- libcrypto 中的 HTTP 或 HTTPS 客户端支持 GET 和 POST 操作、重定向、纯内容和 ASN.1 编码的内容、代理和超时。
- EVP_KDF 使用于密钥派生函数。

- EVP_MAC API 使用 MACs。
- Linux 内核 TLS 支持。

有关更多信息，请参阅 OpenSSL 迁移指南。

网络服务

开源项目 `systemd-networkd` 在现代 Linux 发行版中随处可见。该项目使用的声明式配置语言与其余的 `systemd` 框架类似。它的主要配置文件类型是 `.network` 和 `.link` 文件。

`amazon-ec2-net-utils` 软件包在 `/run/systemd/network` 目录中生成接口特定的配置。当接口连接到实例时，这些配置同时启用接口上的 IPv4 和 IPv6 网络连接。这些配置还安装策略路由规则，有助于确保本地来源的流量通过相应实例的网络接口路由到网络。这些规则可确保从关联的地址或前缀通过弹性网络接口（ENI）路由正确的流量来实现这一点。有关使用 ENI 的更多信息，请参阅《Amazon EC2 用户指南》中的使用 ENI。

您可以通过在 `/etc/systemd/network` 目录中放置一个自定义配置文件以覆盖 `/run/systemd/network` 中包含的默认配置设置，来自定义此网络行为。

systemd.network 文档描述了 `systemd-networkd` 服务如何确定适用于特定接口的配置。它还为 ENI 支持的接口生成替代名称（称为 altnames），以反映各种 AWS 资源的属性。这些 ENI 支持的接口属性是 ENI 连接的 `ENI ID` 和 `DeviceIndex` 字段。您在使用各种工具（例如 `ip` 命令）时，可以使用这些接口的属性来引用这些接口。

AL2023 实例接口名称是使用 `systemd` 插槽命名方案生成的。有关更多信息，请参阅 systemd.net 命名方案。

此外，AL2023 默认使用 `fq_codel` 主动队列管理网络传输调度算法。有关更多信息，请参阅 CoDel 概述。

Core 工具链软件包 glibc、gcc、binutils

Amazon Linux 中的一部分软件包被指定为核心工具链软件包。作为其中的重要组成部分 AL2023，核心软件包将获得五年的支持。我们可能更改某个软件包的版本，但长期支持适用于 Amazon Linux 版本中包含的该软件包。

这三个核心软件包提供用于构建 Amazon Linux 发行版中大多数软件的系统工具链。

软件包	定义	用途
glibc 2.34	系统 C 库	供大多数提供标准函数的二进制程序以及程序和内核之间的接口使用。
gcc 11.2	gcc 编译器套件	编译 C、C++、Fortran。
binutils 2.35	汇编器和链接器以及其他二进制工具	操作或检查二进制程序。

建议在更新任何 glibc 库后进行重启。要更新控制服务的软件包，重新启动服务便足以实现更新。不过，系统重启可以确保之前的所有软件包和库更新都能完成。

软件包管理工具

AL2023 中的默认软件程序包管理工具是 DNF。DNF 是 AL2 中程序包管理工具 YUM 的后续任务。

DNF 的用法类似于 YUM。许多 DNF 命令和命令选项与 YUM 命令相同。在命令行界面 (CLI) 命令中，大多数情况下用 `dnf` 替换 `yum`。

例如，对于以下 AL2 `yum` 命令：

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

在 AL2023 中，它们变为以下命令：

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

在 AL2023 中，虽然 `yum` 命令仍然可用，但作为指针指向 `dnf` 命令。因此，在 shell 或脚本中使用 `yum` 命令时，所有命令和选项都与DNF CLI 相同。有关 YUM CLI 和 DNF CLI 之间的区别的更多信息，请参阅 DNF CLI 相比 YUM 的变化。

有关 `dnf` 命令的命令和选项的完整参考，请参阅手册页 `man dnf`。有关更多信息，请参阅 DNF 命令参考。

默认 SSH 服务器配置

如果您有几年前的 SSH 客户端，可能在连接到实例时会看到一个错误。如果该错误告诉您未找到匹配的主机密钥类型，请更新您的 SSH 主机密钥来解决此问题。

默认禁用 **ssh-rsa** 签名

AL2023 包含一个默认配置，该配置禁用了传统的 ssh-rsa 主机密钥算法并生成一组精简的主机密钥。客户端必须支持 ssh-ed25519 或 ecdsa-sha2-nistp256 主机密钥算法。

该默认配置接受以下任一密钥交换算法：

- 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

默认情况下，AL2023 生成 ed25519 和 ECDSA 主机密钥。客户端支持 ssh-ed25519 或 ecdsa-sha2-nistp256 主机密钥算法。当您通过 SSH 连接到实例时，必须使用支持兼容算法（例如 ssh-ed25519 或 ecdsa-sha2-nistp256）的客户端。如果您需要使用其他密钥类型，请使用用户数据中的一个 cloud-config 片段覆盖生成的密钥列表。

在以下示例中，cloud-config 使用 ecdsa 和 ed25519 密钥生成一个 rsa 主机密钥。

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

如果您使用 RSA 密钥对进行公钥身份验证，则您的 SSH 客户端必须支持 `rsa-sha2-256` 或 `rsa-sha2-512` 签名。如果您使用的是不兼容的客户端并且无法升级，请在您的实例上重新启用 `ssh-rsa` 支持。要重新启用 `ssh-rsa` 支持，请使用以下命令激活 LEGACY 系统加密策略。

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

有关管理主机密钥的更多信息，请参阅 Amazon Linux 主机密钥。

AL2023 中已弃用的功能

在 AL2 中已弃用且不存在于 AL2023 中的功能在此记录。这些功能包括在 AL2 中存在但不在 AL2023 中且不会被添加到 AL2023 的特性和程序包。有关这些功能在 AL2 中支持时长的更多信息，请参阅 AL2 中已弃用的功能。

AL2023 中也存在一些已弃用的功能，这些功能将在未来版本中移除。本章描述这些功能是什么、何时不再受支持以及何时将从 Amazon Linux 中移除。了解已弃用的功能将帮助您部署 AL2023，并为 Amazon Linux 的下一个主要版本做好准备。

主题

compat-软件包

AL2 中任何带有 compat- 前缀的程序包都是为了与尚未针对现代版本程序包重新构建的旧二进制文件保持二进制兼容性而提供的。Amazon Linux 的每个新主要版本都不会沿用之前发布版本中的任何 compat- 程序包。

Amazon Linux 某个发布版（例如 AL2）中的所有 compat- 程序包均已弃用，并且不会出现在后续版本（例如 AL2023）中。我们强烈建议针对更新版本的库重新构建软件。

AL1 中已停用、AL2 中已移除的功能

这部分描述在 AL1 中可用、但在 AL2 中不再可用的功能。

> **Note**
>
> 作为 AL1 维护支持阶段的一部分，某些程序包的生命周期终止日期（EOL）早于 AL1 的生命周期终止日期。更多信息，请参阅 AL1 程序包支持声明。

> **Note**
>
> 部分 AL1 功能在早期版本中已停用。相关信息，请参阅 AL1 发布说明。

主题

32 位 x86（i686）AMI

作为 AL1 2014.09 发布版的一部分，Amazon Linux 宣布这将是最后一个生成 32 位 AMI 的版本。因此，从 AL1 2015.03 发布版开始，Amazon Linux 不再支持在 32 位模式下运行系统。AL2 在 x86-64 主机上为 32 位二进制文件提供有限的运行时支持，且不提供用于构建新 32 位二进制文件的开发程序包。AL2023 不再包含任何 32 位用户空间程序包。我们建议用户在迁移到 AL2023 之前完成向 64 位代码的过渡。

如果需要在 AL2023 上运行 32 位二进制文件，可以在 AL2023 之上运行的 AL2 容器中使用 AL2 的 32 位用户空间。

`aws-apitools-*` 已被 AWS CLI 取代

在 2013 年 9 月 AWS CLI 发布之前，AWS 提供了一组用 Java 实现的命令行实用程序，允许用户进行 Amazon EC2 API 调用。这些工具于 2015 年停用，AWS CLI 成为从命令行与 Amazon EC2 API 交互的首选方式。这组命令行实用程序包括以下 `aws-apitools-*` 程序包。

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

对 `aws-apitools-*` 程序包的上游支持已于 2017 年 3 月结束。尽管缺乏上游支持，Amazon Linux 仍继续提供其中一些命令行实用程序（例如 `aws-apitools-ec2`），以向用户提供向后兼容

性。AWS CLI 是一个比 aws-apitools-* 程序包更稳定可靠、更完整的工具，因为它被积极维护，并提供了使用所有 AWS API 的方法。

aws-apitools-* 程序包已于 2017 年 3 月弃用，将不会收到进一步更新。所有这些程序包的所有用户都应尽快迁移到 AWS CLI。这些程序包在 AL2023 中不存在。

AL1 还提供了 aws-apitools-iam 和 aws-apitools-rds 程序包，这些程序包在 AL1 中已被弃用，并且从 AL2 开始的 Amazon Linux 中不存在。

systemd 在 AL2 中取代 upstart

AL2 是首个使用 systemd 初始化系统的 Amazon Linux 发行版，取代了 AL1 中的 upstart。任何 upstart 特定的配置必须作为从 AL1 迁移到新版 Amazon Linux 的一部分进行更改。无法在 AL1 上使用 systemd，因此从 upstart 迁移到 systemd 只能作为迁移到更新的 Amazon Linux 主要版本（如 AL2 或 AL2023）的一部分来完成。

已在中弃用 AL2 和删除的功能 AL2023

本节介绍中已提供 AL2但中不再提供的功能 AL2023。

主题

- 32 位 x86（i686）程序包
- aws-apitools-*替换为 AWS CLI
- awslogs已弃用，转而使用统一的 Amazon CloudWatch Logs 代理
- bzr 版本控制系统
- cgroup v1
- log4j 热补丁 (log4j-cve-2021-44228-hotpatch)
- lsb_release 和 system-lsb-core 软件包
- mcrypt
- OpenJDK 7 (java-1.7.0-openjdk)
- Python 2.7
- rsyslog-openssl取代 rsyslog-gnutls
- 网络信息服务（NIS）/yp
- Amazon VPC 中的多个域名 create-dhcp-options
- Sun RPC中的glibc

- audit 日志中的 OpenSSH 密钥指纹
- ld.gold 链接器
- ping6
- ftp 程序包

32 位 x86（i686）程序包

作为 2014.09 版本的一部分 AL1，我们宣布这将是最后一个产生 32 位的版本。 AMIs因此，从 2015.03 版本开始， AL1亚马逊 Linux 不再支持在 32 位模式下运行系统。 AL2 为 x86-64 主机上的 32 位二进制文件提供了有限的运行时支持，并且不提供开发包来支持构建新的 32 位二进制文件。 AL2023 不再包含任何 32 位用户空间软件包。我们建议客户完成向 64 位代码的过渡。

如果您需要在上运行 32 位二进制文件 AL2023，则可以在上面运行的 AL2 容器 AL2 内使用 32 位用户空间。 AL2023

`aws-apitools-*`替换为 AWS CLI

在 2013 AWS CLI 年 9 月发布之前，提供 AWS 了一组命令行实用程序，这些实用程序已在中实现 Java，允许客户调用 Amazon EC2 API。这些工具已于 2015 年被弃用， AWS CLI 成为通过命令行与 Amazon EC2 进行交互 APIs 的首选方式。这包括以下 `aws-apitools-*` 程序包。

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

对 `aws-apitools-*` 程序包的上游支持已于 2017 年 3 月结束。尽管缺乏上游支持，Amazon Linux 仍继续提供其中一些命令行实用程序（例如 `aws-apitools-ec2`），以便为客户提供向后兼容性。该 AWS CLI 工具比`aws-apitools-*`软件包更强大、更完整，因为它得到了积极维护，并且提供了一种使用所有软件包的方法 AWS APIs。

`aws-apitools-*` 程序包已于 2017 年 3 月弃用，将不会收到进一步更新。其中任何一个软件包的所有用户都应 AWS CLI 尽快迁移到。中不存在这些软件包 AL2023。

awslogs已弃用，转而使用统一的 Amazon CloudWatch Logs 代理

该awslogs软件包已在中弃用 AL2 ，不再存在于中。 AL2023它已被amazon-cloudwatch-agent软件包中提供的统一 CloudWatch 日志代理所取代。有关更多信息，请参阅 Amazon CloudWatch 日志用户指南。

bzr 版本控制系统

GNU Bazaar(bzr) 版本控制系统已停产 AL2，不再存在于 AL2023。

建议 bzr 用户将其存储库迁移到 git。

cgroup v1

AL2023 移至统一控制组层次结构 (cgroup v2)，而 AL2 使用 cgroup v1。由于 AL2 不支持 cgroup v2，因此需要在迁移到的过程中完成此迁移。 AL2023

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

> **Note**
>
> 该log4j-cve-2021-44228-hotpatch软件包已在中弃用， AL2 并在中删除。 AL2023

为了回应 CVE-2021-44228，亚马逊 Linux 发布了适用于 Ap ache Log4j 的 Hotpatch 的 RPM 打包版本，适用于和。 AL1 AL2在宣布向 Amazon Linux 添加热补丁时，我们指出：“安装热补丁并不能取代更新到可缓解 CVE-2021-44228 或 CVE-2021-45046 的 log4j 版本。”

热补丁是一种缓解措施，可以留出时间来修补 log4j。的第一个正式发布版本 AL2023 是 CVE-2021-44228 发布的 15 个月后，因此 AL2023 不附带该热补丁（无论是否启用）。

建议在 Amazon Linux 上运行自己的 log4j 版本的客户务必更新到未受 CVE-2021-44228 或 CVE-2021-45046 影响的版本。

lsb_release 和 system-lsb-core 软件包

过去，有些软件会调用该lsb_release命令（ AL2 由软件system-lsb-core包提供）来获取有关其运行的 Linux 发行版的信息。Linux Standards Base (LSB) 引入了此命令，Linux 发行版采用了此命令。Linux 发行版已经演变为使用更简单的标准，将这些信息保存在 /etc/os-release 和其他相关文件中。

os-release 标准来自 systemd。有关更多信息，请参阅 systemd os-release 文档。

AL2023 不随lsb_release命令一起提供，也不包括system-lsb-core软件包。软件应完成向 os-release 标准的过渡，以保持与 Amazon Linux 和其他主要 Linux 发行版的兼容性。

mcrypt

该mcrypt库和相关PHP扩展已在中弃用 AL2，不再存在于中。 AL2023

上游 PHP 在 PHP 7.1 中弃用了 mcrypt 扩展，该版本最初于 2016 年 12 月发布，最终版本于 2019 年 10 月发布。

上游mcrypt库最后一次发布是在 2007 年，并未在 2017 年完成新提交SourceForge 所需的cvs版本控制迁移，最近一次提交（之前只有 3 年）是从 2011 年开始的，删除了该项目有维护者的提法。

建议所有剩余mcrypt的用户将其代码移植到OpenSSL，因为mcrypt不会添加到 AL2023。

OpenJDK 7 (java-1.7.0-openjdk)

> **Note**
>
> AL2023 提供了多个版本的 Amazon Corretto 来支持基于的工作负载。JavaOpenJDK 7 软件包已 AL2在中弃用，并且不再存在于中。 AL2023目前可用的最古老 AL2023 的 JDK 由 Corretto 8 提供。

有关 Amazon Linux 上 Java 的更多信息，请参阅 AL2023 中的 Java。

Python 2.7

> **Note**
>
> AL2023 移除了 Python 2.7，因此任何需要 Python 的操作系统组件都是为了与 Python 3 配合使用而编写的。要继续使用 Amazon Linux 提供并支持的 Python 版本，请将 Python 2 代码转换为 Python 3。

有关 Amazon Linux 上 Python 的更多信息，请参阅 AL2023 中的 Python。

rsyslog-openssl取代 rsyslog-gnutls

该rsyslog-gnutls软件包已在中弃用 AL2，不再存在于中。 AL2023rsyslog-openssl 程序包应能直接替代 rsyslog-gnutls 程序包的所有用途。

网络信息服务（NIS）/yp

网络信息服务 (NIS)，最初称为黄页，或者YP已在中弃用 AL2，不再存在于中 AL2023。这包括以下程序包：ypbind、ypserv 和 yp-tools。与之集成的其他软件包NIS已在中删除了此功能 AL2023。

Amazon VPC 中的多个域名 create-dhcp-options

在 Amazon Linux 2 中，可以在 domain-name 参数中向 create-dhcp-options 传递多个域名，这将导致 /etc/resolv.conf 包含类似 search foo.example.com bar.example.com 的内容。Amazon VPC DHCP 服务器使用 DHCP 选项 15 发送提供的域名列表，该选项仅支持单个域名（参阅 RFC 2132 第 3.17 节）。由于 AL2023 systemd-networkd用于网络配置（如下所示）RFC，因此中 AL2 不存在此意外功能 AL2023

AWS CLI 和 Amazon VPC 文档对此说明如下：“某些 Linux 操作系统接受以空格分隔的多个域名。但是，Windows 以及其他 Linux 操作系统将该值视为单个域，因而会导致意外行为。如果您的 DHCP 选项集与其中实例所运行操作系统将该值视为单个域的 Amazon VPC 关联，请仅指定一个域名。”

在这些系统上 AL2023，例如使用DHCP选项 15（仅允许一个）指定两个域，并且由于域名中的空格字符无效，这将导致空格字符被编码为032，从而导致/etc/resolv.conf包含search foo.exmple.com032bar.example.com。

为支持多个域名，DHCP 服务器应使用 DHCP 选项 119（参阅 RFC 3397 第 2 节）。有关 Amazon VPC DHCP 服务器何时支持此功能的信息，请参阅《Amazon VPC 用户指南》。

Sun RPC中的glibc

in 的实现已Sun RPC在中弃glibc用， AL2 并在中删除。 AL2023如果需要Sun RPC功能，建议客户转而使用该libtirpc库（在 AL2 和中提供 AL2023）。采用 libtirpc 还能使应用程序支持 IPv6。

此变更反映了更广泛社区对上游 glibc 移除此功能的采纳，例如 Fedora 从 glibc 中移除 Sun RPC 接口以及 Gentoo 中的类似变更。

audit 日志中的 OpenSSH 密钥指纹

在生命周期的后期 AL2，在 OpenSSH 包中添加了一个补丁，用于发出用于身份验证的密钥指纹。中不存在此功能 AL2023。

ld.gold 链接器

ld.gold链接器在中可用 AL2，并且已在中删除。 AL2023构建明确引用 gold 链接器的软件的客户应迁移至常规 (ld.bfd) 链接器。

上游 GNU Binutils 的 2.44 版（2025 年 2 月发布）发布说明记录了 ld.gold 的移除："与我们以往做法不同，在此版本中，binutils-2.44.tar 压缩包不包含 gold 链接器的源代码。这是因为 gold 链接器现已弃用，除非有志愿者站出来愿意继续开发和维护，否则最终将被移除。"

ping6

在中 AL2023，常规ping实用程序原生支持 IPv6，/bin/ping6不再需要分开。中 AL2023，/usr/sbin/ping6是指向/usr/bin/ping可执行文件的符号链接。

这一变化是在更广泛的社区采用提供此功能的较新iputils版本之后进行的，例如 Fedora 中的 Ping IPv6 更改。

ftp 程序包

从 AL2023 开始，AL2 中的 ftp 程序包在 Amazon Linux 中不再提供。此决定是我们对安全性、可维护性和现代软件开发实践持续承诺的一部分。在迁移至 AL2023 的过程中（或之前），我们建议将任何对传统 ftp 程序包的使用迁移至其替代方案之一。

背景

传统的 ftp 程序包在上游已多年未得到积极维护。其源代码的最后一次重要更新发生在 2000 年代初期，且原始源代码存储库已不可用。尽管一些 Linux 发行版提供了安全漏洞补丁，但该代码库基本上仍处于无人维护状态。

建议的替代方案

AL2023 为 FTP 功能提供了几种现代的、积极维护的替代方案：

lftp（在 AL2 和 AL2023 中可用）

一个复杂的文件传输功能程序，支持 FTP、HTTP、SFTP 和其他协议。它比传统的 ftp 客户端提供更多功能，并且得到积极维护。

安装命令：dnf install lftp。

`curl`（在 AL2 和 AL2023 中可用）

一个多功能命令行工具，用于通过 URL 传输数据，支持 FTP、FTPS、HTTP、HTTPS 和许多其他协议。

在 AL2023 中通过 `curl-minimal` 程序包默认提供。如需更广泛的协议支持，可选择使用 `curl-full` 升级至 dnf swap curl-minimal curl-full。

`wget`（在 AL2 和 AL2023 中可用）

一个用于从网络下载文件的非交互式命令行实用程序，支持 HTTP、HTTPS 和 FTP 协议。

安装命令：dnf install wget（未在所有 AL2023 映像中默认安装）

`sftp`（在 AL2 和 AL2023 中可用）

一种通过 SSH 运行的安全文件传输功能协议，提供加密的文件传输。

作为 OpenSSH 程序包的组成部分默认提供。

迁移注意事项

如果您的应用程序或脚本依赖传统的 `ftp` 客户端，请考虑以下迁移方法：

1. 更新脚本以使用现代替代方案：修改您的脚本以使用 `lftp`、`curl`、`wget`、或 `sftp` 替代传统的 `ftp` 客户端。
2. 检查程序包依赖关系：某些应用程序可能在其程序包元数据中将 `ftp` 程序包列为依赖项，即使它们内部早已迁移使用现代协议。在这些情况下，即使缺少 `ftp` 程序包中的 `/usr/bin/ftp`，应用程序仍可能在 AL2023 上正常运行。请检查应用程序的实际要求，而非仅依赖声明的依赖关系。
3. 更新应用程序依赖关系：对于您维护的仍声明依赖 `ftp` 程序包但实际并未使用的应用程序，请更新程序包元数据以移除此不必要的依赖项。

安全考虑因素

FTP 协议以明文形式传输数据，包括身份验证凭证。对于安全敏感型应用程序，我们强烈建议使用加密替代方案（例如 SFTP 或 HTTPS），这些方案均受推荐替代工具的支持。

AL2023 中已弃用的功能

这部分描述 AL2023 中存在但很可能在 Amazon Linux 未来版本中移除的功能。每节将描述该功能是什么以及预计何时会从 Amazon Linux 中移除。

> **Note**
>
> 随着 Linux 生态系统的发展以及未来 Amazon Linux 主要版本临近发布，这部分将随时间推移而更新。

主题

32 位 x86（i686）运行时支持

AL2023 保留了运行 32 位 x86（i686）二进制文件的能力。Amazon Linux 的下一个主要版本很可能将不再支持运行 32 位用户空间二进制文件。

`aspell`

虽然 AL2023 随附了 `aspell` 程序包，但它已被弃用，并将在 Amazon Linux 的下一个主要发布版中移除。建议客户迁移到现代替代方案，例如 `hunspell` 或 `enchant2`。

AL2023 中对 `aspell` 的弃用遵循了更广泛的社区转变，例如 `aspell` 在 Fedora 中的弃用。

Berkeley DB (`libdb`)

AL2023 随附 Berkeley DB (`libdb`) 库的 5.3.28 版本。这是在许可证从限制较少的 Sleepycat 许可证变更为 GNU Affero GPLv3（AGPL）许可证之前的最后一个 Berkeley DB 版本。

AL2023 中很少有程序包仍依赖 Berkeley DB (`libdb`)，该库将在 Amazon Linux 的下一个主要发布版中移除。

> **Note**
>
> AL2023 中的 dnf 程序包管理器保留了对 Berkeley DB（BDB）格式 rpm 数据库的只读支持。此支持将在 Amazon Linux 的下一个主要发布版中移除。

对 libdb 的弃用遵循了更广泛社区远离该技术的趋势，例如在 Fedora 中的 libdb 弃用。

cron

默认情况下，cronie 软件包安装在 AL2 AMI 上，为计划定期任务的传统 crontab 方式提供支持。在 AL2023 中，默认不包含 cronie。因此，默认不再提供对 crontab 的支持。

在 AL2023 中，您可以选择安装 cronie 程序包以使用传统的 cron 作业。由于 systemd 提供附加功能，建议您迁移到 systemd 计时器。

Amazon Linux 的未来版本（可能是下一个主要版本）可能不再包含对传统 cron 作业的支持，并完成向 systemd 计时器的过渡。我们建议您停止使用 cron。

IMDSv1

默认情况下，AL2023 AMI 配置为以仅 IMDSv2 模式启动，禁用 IMDSv1 的使用。仍可选择启用 IMDSv1 使用 AL2023。Amazon Linux 的未来版本很可能将强制实施仅 IMDSv2 模式。

有关 AMI 的 IMDS 配置的更多信息，请参阅《Amazon EC2 用户指南》中的配置 AMI。

pcre 版本 1

传统的 pcre 程序包已弃用，并将在 Amazon Linux 的下一个主要发布版中移除。pcre2 软件包是后继版本。尽管 AL2023 的初始版本随附了数量有限的针对 pcre 构建的程序包，但这些程序包将在 AL2023 版本内迁移至 pcre2。已弃用的 pcre 库在 AL2023 中仍将可用。

> **Note**
>
> 已弃用版本的 pcre 在 AL2023 的整个生命周期内将不会收到安全更新。有关 pcre 支持生命周期以及该程序包将接收安全更新时长的更多信息，请参阅关于 pcre 程序包的程序包支持声明。

弃用 pcre 而转向 pcre2 遵循了更广泛社区朝此方向的转变，例如在 Fedora 中 pcre 的弃用。

System V init (`sysvinit`)

尽管 AL2023 保留了对 System V 服务 (init) 脚本的向后兼容性，但上游 `systemd` 项目在其 v254 发布版中宣布了对 System V 服务脚本支持的弃用，并表明将在未来的 `systemd` 版本中移除该支持。有关更多信息，请参阅 systemd。

AL2023 将保留对 System V 服务（init）脚本的向后兼容性，但鼓励用户迁移到使用原生 `systemd` 单元文件，以便为 Amazon Linux（很可能在下一个主要发布版中）移除对 System V 服务（init）脚本的支持做好准备。

已终止生命周期的程序包被弃用

AL2023 中可用的每个程序包都有一个相关的支持声明，其中包含 Amazon Linux 特定信息。这些声明涵盖了操作系统的核心及其生命周期，以及诸如 the section called “PHP” 和 the section called “Python” 之类的程序包，对于这些程序包，AL2023 会提供多个版本，并且每个版本在上游开源项目支持的期限内都会得到支持。

在 AL2023 中，您可以使用 `dnf` 程序包管理器获取程序包支持信息。有关更多信息，请参阅 获取程序包支持信息。

如果某个程序包在 Amazon Linux 主要版本结束前不再受支持，则应假定该程序包已被弃用，并且不会出现在 Amazon Linux 的下一个主要版本中。

对于诸如 the section called “PHP” 和 the section called “Python” 之类的程序包，每个 Amazon Linux 主要版本都提供了多个版本，且每个版本具有不同的支持生命周期，这些程序包很可能将继续存在于新的 Amazon Linux 主要版本中，尽管程序包的主要版本之间几乎没有或完全没有重叠。建议在选择依赖项时牢记 Amazon Linux 程序包支持时间线。

比较 AL2 和 AL2 023

以下主题描述了 AL2 和 AL2 023 之间的主要区别。

有关、和 AL2 023 中已弃用的功能的更多信息 AL1 AL2，请参阅。AL2023 中已弃用的功能

主题

- 添加、升级和删除了软件包
- 针对每个版本的支持
- 命名和版本控制更改
- 优化
- 来自多个上游
- 网络系统服务
- 软件包管理器
- 使用 cloud-init
- 图形桌面支持
- 编译器三元组
- 32 位 x86 (i686) 软件包
- lsb_release 和 system-lsb-core 软件包
- Extra Packages for Enterprise Linux (EPEL)
- Python 2.7 已被 Python 3 所取代
- 安全更新
- 确定性升级用于提高稳定性
- gp3 作为默认 Amazon EBS 卷类型
- 统一控制组层次结构 (cgroup v2)
- systemd 计时器取代 cron
- 改进的工具链：gcc、binutils 和 glibc
- systemd 日志取代 rsyslog
- 最小化程序包依赖
- Amazon Corretto 作为默认 JVM

- AWS CLIv2
- UEFI 首选和安全启动
- SSH 服务器默认配置更改
- AL2023 内核更改自 AL2
- /tmp 现在是 tmpfs
- AMI 和容器映像变化
- 比较 Amazon Linux 2 和 Amazon Linux 2023 AMI 上安装的软件包
- 比较 Amazon Linux 2 和 Amazon Linux 2023 Minimal AMI 上安装的软件包
- 比较 Amazon Linux 2 和 Amazon Linux 2023 基础容器映像上安装的软件包

添加、升级和删除了软件包

AL2023 包含数千个可供使用的软件包。有关 023 年新增、升级或删除的所有软件包与之前的 Amazon Linux 版本对比的完整列表，请参阅 AL2 023 年 AL2的 Package 变更。

要申请在 AL2 023 年添加或更改软件包，请在亚马逊 linux- 2023 存储库中提交问题。 GitHub

针对每个版本的支持

对于 AL2 023，我们提供五年的支持。

有关更多信息，请参阅 发布频率。

命名和版本控制更改

AL2023 支持的机制与支持平台识别的 AL2 机制相同。 AL2023 还引入了用于平台识别的新文件。

有关更多信息，请参阅 命名和版本控制。

优化

AL2023 优化了启动时间，以缩短从实例启动到运行客户工作负载的时间。这些优化涵盖了 Amazon EC2 实例内核`cloud-init`配置、配置和内置于操作系统软件包（如`kmod`和`systemd`）中的功能。

有关优化的更多信息，请参阅性能和操作优化。

来自多个上游

AL2023 基于 RPM，包括来自多个版本的 Fedora 和其他发行版（例如 CentOS 9 Stream）的组件。Amazon Linux 内核直接源自来自 kernel.org 的长期支持 (LTS) 版本，该版本是独立于其他发行版选择的。

有关更多信息，请参阅 与 Fedora 的关系。

网络系统服务

`systemd-networkd`系统服务在 AL2 023 中管理网络接口。这是从 AL2、使用ISC dhclient或的改动`dhclient`。

有关更多信息，请参阅 网络服务。

软件包管理器

AL2023 上的默认软件包管理工具是DNF。 DNF是中的软件包管理工具的继任者 AL2。YUM

有关更多信息，请参阅 软件包管理工具。

使用 cloud-init

在 AL2 023 中，cloud-init管理软件包存储库。默认情况下，在早期版本的Amazon Linux 中，cloud-init安装了安全更新。这不是 AL2 023 的默认设置。用于在启动时更新的确定性升级功能描述了 AL2 023 `releasever` 在启动时启用软件包更新的方法。有关更多信息，请参阅在中管理软件包和操作系统更新 AL2023和确定性升级用于提高稳定性。

对 AL2于 023，你可以cloud-init与一起使用。SELinux有关更多信息，请参阅 使用 cloud-init 启用 `enforcing` 模式。

Cloud-init 使用 cloud-init 从使用 HTTP(S) 的远程位置加载配置内容。在早期版本中，当远程资源不可用时，Amazon Linux 不会提醒您。在 AL2 023 中，不可用的远程资源会造成致命错误并导致cloud-init执行失败。此行为更改从 AL2，提供了更安全的 “失效关闭” 默认行为。

有关更多信息，请参阅自定义 cloud-init 和 cloud-init 文档。

图形桌面支持

AL2自 2023.7 版本起，023 采用基于 GNOME 的图形桌面环境，取代了中使用的 MATE 桌面。 AL2 此版本为用户提供了不同的桌面体验，同时保持了 AL2 023 的云优化性能。GNOME 桌面环境提供多种自定义选项、系统集成功能以及独特的用户界面设计，为用户提供了不同于先前 MATE 桌面环境的替代方案。更多详细信息请参阅 AL2023 图形化桌面 页面。

编译器三元组

AL2023 为和设置编译器三元组GCC，LLVM以表明amazon这是供应商。

因此， AL2 aarch64-redhat-linux-gcc变成 aarch64-amazon-linux-gcc AL2 023。

对于大多数用户来说，这应该是完全透明的，并且可能只会影响那些在 AL2 023 上构建编译器的用户。

32 位 x86 (i686) 软件包

作为 2014.09版本的一部分， AL1它宣布这将是最后一个生产32位的版本。 AMIs因此，从 2015.03 版本起， AL1Amazon Linux 不再支持在 32 位模式下运行系统。 AL2 为 x86-64 主机上的 32 位二进制文件提供了有限的运行时支持，并且没有提供支持构建新 32 位二进制文件的开发包。 AL2023 不再包含任何 32 位用户空间软件包。我们建议您完成向 64 位代码的过渡。

如果您需要在 023 上运行 32 位二进制文件，则可以从 AL2 023 之上运行的 AL2 容器 AL2 内部使用 32 位用户空间。 AL2

lsb_release 和 system-lsb-core 软件包

过去，有些软件会调用该lsb_release命令（ AL2 由软件system-lsb-core包提供）来获取有关其运行的 Linux 发行版的信息。Linux Standards Base (LSB) 引入了此命令，Linux 发行版采用了此命令。Linux 发行版已经演变为使用更简单的标准，将这些信息保存在 /etc/os-release 和其他相关文件中。

os-release 标准来自 systemd。有关更多信息，请参阅 systemd os-release 文档。

AL2023 不附带该lsb_release命令，也不包括system-lsb-core软件包。软件应完成向 os-release 标准的过渡，以保持与 Amazon Linux 和其他主要 Linux 发行版的兼容性。

Extra Packages for Enterprise Linux (EPEL)

> **Warning**
>
> E AL2 epel xtra 启用了第三方EPEL7存储库。自 2024 年 6 月 30 日起，不再维护第三方 EPEL7 存储库。
> 此第三方存储库未来将不再更新。这意味着 EPEL 存储库中的程序包将不会有安全修复。
> 本节将介绍 AL2 023 中包含的软件包的选项。EPEL

Extra Packages for Enterprise Linux (EPEL) 是 Fedora 社区中的一个项目，其目标是为企业级 Linux 操作系统创建大量软件包。该项目主要制作RHEL和CentOS包装。 AL2 具有高度的兼容性CentOS 7。因此，许多EPEL7软件包都能正常运行 AL2。

没有与 AL2 023 二进制兼容的EPEL版本。但是，想要在 AL2 023 年使用其EPEL7套餐的客户有几种选择。有些EPEL软件包在 AL2 023 中有替代方案，而另一些则作为其中的一部分适用于亚马逊 Linux 的补充软件包提供。

> **Warning**
>
> 仅添加设计用于 AL2 023 的存储库。
> 虽然为其他发行版设计的存储库现在可以使用，但不能保证它们会继续使用 023 中的任何软件包更新或不是为在 AL2 023 中使用而设计的存储库。 AL2

本页提供有关客户在 AL2 023 上使用的EPEL7软件包 AL2 及其对应套餐的信息。

对于其余套餐，买家或许可以使用适用于亚马逊 Linux 的补充套餐 (SPAL)。SPAL 提供了数EPEL9千个专为 Amazon Linux 2023 构建的软件包，但这些软件包不在AWS支持计划范围内。这意味着 CVEs 不会对 SPAL 软件包进行跟踪，并且只有在上游可用时才会提供补丁。

> **Important**
>
> 使用适用于亚马逊 Linux 的补充软件包前请查阅的文档。

主题

- axel- HTTP/FTP 客户

- brotli 和 libbrotli：压缩
- collectd：统计数据收集进程守护程序
- cpulimit：CPU 使用限制器
- exim：邮件传输代理
- fuse3：用户空间文件系统（FUSE）v3
- ganglia：分布式监控系统
- git-lfs：使用 Git 进行大文件版本控制
- haveged：使用 HAVEGE 算法的熵源
- inotify-tools：inotify 命令行工具
- iperf- TCP/UDP 性能基准
- jemalloc：替代的 malloc 实现
- libbsd：BSD 兼容函数库
- libserf - HTTP 客户端库
- libzstd：zstd 压缩库
- lighttpd Web 服务器
- lshell：受限 Shell
- monit：进程、文件、目录和设备监视器
- nodejs
- perl-Config-General
- python2-lockfile：文件锁定
- python2-rsa：纯 Python RSA
- python2-simplejson：适用于 Python 2 的 JSON 例程
- rkhunter：Rootkit 检测工具
- rssh：与 OpenSSH 配合使用的受限 Shell
- sscg：自签名 SSL 证书生成器
- stress：压力测试工具
- stress-ng：压力测试工具
- tmpwatch：基于最后访问时间删除文件
- xmlstarlet：命令行 XML 工具集

axel- HTTP/FTP 客户

axel 程序包包含在 EPEL7 中，且从未作为 Amazon Linux 的一部分发布。 AL2023 中可用的替代方案是curl和。wget

> **Warning**
>
> axel 的 -S 选项使用未加密的 http 连接来发现文件的镜像。

强烈建议将所有对 axel 的使用迁移到 curl 或 wget。

brotli 和 libbrotli：压缩

brotli和libbrotli软件包在里EPEL7面，而 AL2 核心中只有brotli软件包可用。

brotli和libbrotli软件包都包含在 AL2 023 中。

可以使用以下命令在 AL2 023 上安装该brotli软件包：

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

可以使用以下命令在 AL2 023 上安装该libbrotli软件包：

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

collectd：统计数据收集进程守护程序

该collect软件包已经在里面了EPEL7，还有 E collectd-python3 AL2 xtra collectd s 中有。

该collectd软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

cpulimit：CPU 使用限制器

在 Amazon Linux 2023 中，systemd 提供限制进程或进程组 CPU 使用量的功能。此功能也可轻松地在任何 systemd 服务中使用。

systemd 提供了强大的资源控制功能，可用于确保对任何任务或任务组可以使用的资源进行限制。有关更多信息，请参阅上游 systemd.resource-control 文档以及在 AL2023 中使用 systemd 限制进程资源使用量。

exim：邮件传输代理

该exim软件包已在里面EPEL7，之前有售 AL1。亚马逊 Linux 2023 同时提供postfix和sendmail邮件传输代理 (MTAs)。

fuse3：用户空间文件系统（FUSE）v3

fuse3 程序包（包括 fuse3-libs 和 fuse3-devel）包含在 EPEL7 中。这些软件包是 AL2 023 的一部分，每个软件包都可以通过运行以下相关命令进行安装：

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

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

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

ganglia：分布式监控系统

该ganglia软件包已在里面EPEL7，之前有售 AL1。它不是随附的 AL2。

上游项目有一段不活跃的时期，有些未解决的问题没有 CVEs得到解决。尽管上游项目最近有一些活动，但不计划将其增加ganglia到 AL2 023。

git-lfs：使用 Git 进行大文件版本控制

git-lfs 程序包包含在 EPEL7 中。在 Amazon Linux 2023 中，git-lfs 程序包包含在核心存储库中。在 AL2 023 上，git-lfs可以通过运行以下命令进行安装：

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

haveged：使用 HAVEGE 算法的熵源

haveged 程序包包含在 EPEL7 中。Amazon Linux 2023 预配置了熵源，无需使用 haveged。

inotify-tools：inotify 命令行工具

inotify-tools包裹已装入EPEL7，并包含在 AL2 023 中。

> **Note**
>
> 在 AL2 023 中，systemd支持基于路径的激活，可用于对事件（例如路径存在或更改时）采取行动。
> 现在可以使用 systemd 路径激活以更可靠的方式更好地完成 inotify-tools 的许多用途。有关更多信息，请参阅 systemd.path。

该inotify-tools软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

iperf- TCP/UDP 性能基准

第 2 iperf 版软件包已testing AL2 推EPEL7出，也在 Extra 中可用。还有 AL1

> **Note**
>
> iperf3 程序包也可用，提供 iperf 的第 3 版。

该iperf软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

jemalloc：替代的 malloc 实现

该jemalloc软件包已装EPEL7入，并在 E mariadb10.5 AL2 xtra lamp-mariadb10.2-php7.2 s 中提供。

该jemalloc软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

libbsd：BSD 兼容函数库

该libbsd软件包已装入EPEL7，也可在 E testing AL2 xtra 中购买。

该libbsd软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

可通过运行以下命令安装 libbsd 的开发文件。

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

libserf - HTTP 客户端库

libserf 程序包包含在 EPEL7 中。libserf 程序包在 Amazon Linux 2023 中提供。可通过运行以下命令安装：

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

libzstd：zstd 压缩库

该libzstd软件包在 AL2 核心和内核中EPEL7。该libzstd软件包也是 AL2 023 的一部分。

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

lighttpd Web 服务器

该lighttpd软件包已在里面EPEL7，之前有售 AL1。Amazon Linux 2023 同时提供 Apache httpd 和 nginx Web 服务器。

lshell：受限 Shell

lshell 程序包从未作为 Amazon Linux 的一部分发布。它曾在 EPEL6 中可用。lshell 的 Fedora 打包存储库说明了在 EPEL7 或 Fedora 30 中未将其打包的原因。它也已从 Debian 中移除。

上游lshell项目已停止积极维护，并且包含已知的未修补的 Critical CVEs：CVE-2016-6902 和 CVE-2016-6903。

在 Debian 错误中建议的替代方案 rssh 其上游也已无人维护，作者列举了无法修复的安全问题作为原因。

出于这些原因，没有计划添加lshell到 AL2 023。

monit：进程、文件、目录和设备监视器

在 Amazon Linux 2023 中，systemd 提供了用于监控、启动、停止和重启服务的大量功能。这包括限制重启频率、在重启尝试之间等待，以及在失败时启动其他服务。更多信息，请参阅 systemd.service 文档。

在 AL2 023 中，systemd还支持基于路径的激活，可用于对事件（例如路径存在或更改时）采取行动。更多信息，请参阅 systemd.path。

systemd 单元具有通用配置选项，可用于指定依赖关系、条件判断以及成功或失败时执行的操作。更多信息，请参阅 systemd.unit 文档。

systemd 提供了强大的资源控制功能，可用于确保任何监控任务不会占用过多 CPU 或内存。更多信息，请参阅 systemd.resource-control。

nodejs

nodejs版本 16 软件包已包含在 023 中EPEL7，nodejs现在已包含在 AL2 023 中。在撰写本文时，nodejs版本 18 和 20 都已在 AL2 023 年推出。你可以使用以下命令在 AL2 023 上安装 nodejs 18：

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

你可以使用以下命令在 AL2 023 上安装 nodejs 20：

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

perl-Config-General

该perl-Config-General软件包已包含在 023 中EPEL7，现在已包含在 AL2 023 中。你可以使用以下命令在 AL2 023 中安装perl-Config-General软件包：

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

也可通过要求 DNF 安装提供特定 Perl 模块的程序包来安装 Perl 模块。通过此方法，您可使用更熟悉的 Perl 模块名称而非操作系统程序包名称。

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

python2-lockfile：文件锁定

python2-lockfile包裹在里面EPEL7，里面有一个 AL2 包python-lockfile裹。在 AL2 023 中Python 2.7 已被 Python 3 所取代，因此该软件包的 Python 2 变体不会添加到 AL2 023 中。

此软件包的 Python 3 版本包含在 AL2 023 中。您可以使用以下命令之一在 AL2 023 中安装该python3-lockfile软件包：

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

也可以通过要求 DNF 安装提供特定 Python 模块的程序包来安装 Python 模块。

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

python2-rsa：纯 Python RSA

python2-rsa包裹在里面EPEL7，里面有一个 AL2 包python2-rsa裹。在 AL2 023 中Python 2.7 已被 Python 3 所取代，因此该软件包的 Python 2 变体不会添加到 AL2 023 中。

此软件包的 Python 3 版本包含在 AL2 023 中。您可以使用以下命令之一在 AL2 023 中安装该python3-rsa软件包：

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

也可以通过要求 DNF 安装提供特定 Python 模块的程序包来安装 Python 模块。

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

python2-simplejson：适用于 Python 2 的 JSON 例程

python2-simplejson 程序包包含在 EPEL7 中。在 AL2 023 中Python 2.7 已被 Python 3 所取代，因此该软件包的 Python 2 变体不会添加到 AL2 023 中。

此软件包的 Python 3 版本包含在 AL2 023 中。你可以使用以下命令在 AL2 023 中安装python3-simplejson软件包：

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

也可以通过要求 DNF 安装提供特定 Python 模块的程序包来安装 Python 模块。

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

rkhunter：Rootkit 检测工具

该rkhunter软件包随附在 AL2 023 中。chkrootkit

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

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

rssh：与 OpenSSH 配合使用的受限 Shell

rssh 程序包包含在 EPEL7 中。上游 rssh 程序包已无人维护，作者列举了无法修复的安全问题作为原因。

由于作者列举了无法修复的安全问题，因此没有计划添加rssh到 AL2 023。

sscg：自签名 SSL 证书生成器

该sscg软件包在 AL2 核心和内核中EPEL7。该sscg软件包也是 AL2 023 的一部分。

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

stress：压力测试工具

stress包裹已装入EPEL7，也有 AL1

该stress软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

stress-ng：压力测试工具

该stress-ng软件包已装入EPEL7，也可在 E testing AL2 xtra 中购买。

该stress-ng软件包包含在 AL2 023 中，可以通过运行以下命令进行安装：

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

tmpwatch：基于最后访问时间删除文件

在 Amazon Linux 2023 中，此功能由 systemd-tmpfiles 提供。

xmlstarlet：命令行 XML 工具集

该xmlstarlet软件包已在 AL2 023 年上市EPEL7，但尚未上市。

上游程序包已超过 9 年未更新（最后更新于 2014 年 8 月）。在那之前的四年期间（至少从 2010 年 7 月起），寻找新维护者的请求始终无人回应。正是出于这个原因，不打算将其添加xmlstarlet到 AL2 023。

Python 2.7 已被 Python 3 所取代

AL2 在 2025 年 6 月之前为 Python 2.7 提供支持和安全补丁，这是我们对 AL2 核心包的长期支持 (LTS) 承诺的一部分。这种支持不仅限于上游 Python 社区在 2020 年 1 月发布的 Python 2.7 end-of-life 声明。

AL2 使用yum包管理器，它对 Python 2.7 有硬依赖性。在 AL2 023 中，dnf包管理器已迁移到 Python 3，不再需要 Python 2.7。 AL2023 已完全移至 Python 3。

> **Note**
>
> AL2023 删除了 Python 2.7，因此任何需要 Python 的操作系统组件都是为了与 Python 3 配合使用而编写的。要继续使用 Amazon Linux 提供并支持的 Python 版本，请将 Python 2 代码转换为 Python 3。

有关 Amazon Linux 上的 Python 的更多信息，请参阅 AL2023 中的 Python。

安全更新

亚马逊 Linux 2023 在中存在的强化基础上进行了改进。 AL2有关更多信息，请参阅 Amazon Linux 2023 中的安全性与合规性。有关中内核强化更改的更多信息 AL2，请参阅注重安全的内核配置变化。

主题

SELinux

默认情况下，f Security Enhanced Linux (SELinux) o AL2 r 023 为enabled并设置为permissive模式。在 permissive 模式下，记录但不强制执行权限拒绝。

SELinux是 Amazon Linux 内核的一项安全功能，该内核曾disabled在 AL2。 SELinux是内核功能和实用程序的集合，为内核的主要子系统提供强制访问控制 (MAC) 架构。

有关更多信息，请参阅 为 AL2023 设置 SELinux 模式。

有关SELinux存储库、工具和策略的更多信息，请参阅SELinux 笔记本、 SELinux 策略类型和SELinux 项目。

OpenSSL 3

AL2023 具有Open Secure Sockets Layer version 3 (OpenSSL 3)密码学工具包。 AL2023 支持TLS 1.3和TLS 1.2网络协议。

默认情况下， AL2 附带OpenSSL 1.0.2。您可以依据 OpenSSL 1.1.1 构建应用程序。

有关 OpenSSL 的更多信息，请参阅 OpenSSL 迁移指南。

有关安全性的更多信息，请参阅安全更新和功能。

IMDSv2

默认情况下，任何使用 AL2 023 AMI 启动的实例都IMDSv2只需要-并且您的默认跳数限制将设置为 2 以支持容器化工作负载。这可通过将 imds-support 参数设置为 v2.0 来完成。有关更多信息，请参阅 Amazon EC2 用户指南中的配置 AMI。

> **Note**
>
> 会话令牌的有效时间可以介于 1 秒到 6 小时之间。用于发送针对 IMDSv2 查询的 API 请求的地址如下：

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

您可以手动覆盖这些设置并使用实例元数据选项启动属性启用 IMDSv1。此外，您可以使用 IAM 控件强制执行不同的 IMDS 设置。有关设置和使用实例元数据服务的更多信息，请参阅 Amazon EC2 用户指南中的使用IMDSv2、为新实例配置实例元数据选项和修改现有实例的实例元数据选项。

删除 log4j 热补丁 (`log4j-cve-2021-44228-hotpatch`)

> **Note**
>
> AL2023 不随`log4j-cve-2021-44228-hotpatch`包裹一起发货。

为了回应 CVE-2021-44228，亚马逊 Linux 发布了适用于 Ap ache Log4j 的 Hotpatch 的 RPM 打包版本，适用于和。 AL1 AL2在宣布向 Amazon Linux 添加热补丁时，我们指出：“安装热补丁并不能取代更新到可缓解 CVE-2021-44228 或 CVE-2021-45046 的 log4j 版本。”

热补丁是一种缓解措施，可以留出时间来修补 `log4j`。 AL2023 的第一个通用可用性 (GA) 版本是 CVE-2021-44228 发布的 15 个月后，因此 AL2 023 不附带热补丁（无论是否启用）。

在 Amazon Linux 上运行自有 `log4j` 版本的用户应确保已更新到不受 CVE-2021-44228 或 CVE-2021-45046 影响的版本。

AL2023 提供了相关指导，正在更新 AL2023以便您可以及时了解最新的安全补丁。安全通告发布在 Amazon Linux 安全中心上。

确定性升级用于提高稳定性

借助通过版本控制存储库进行确定性升级功能，默认情况下，每个 AL2 023 AMI 都锁定到特定的存储库版本。您可以使用确定性升级来提高软件包版本和更新之间的一致性。每个版本，无论是主要版本还是次要版本，都包含一个特定的存储库版本。

AL2023 的新增功能，默认启用确定性升级。这是对和其他早期版本中 AL2 使用的手动增量锁定方法的改进。

有关更多信息，请参阅 通过版本控制的存储库进行确定性升级 AL2023。

gp3 作为默认 Amazon EBS 卷类型

AL2023 AMI 和 AL2 两者都使用根XFS文件系统上的文件系统。对于 AL2 023，根设备文件系统的mkfs选项已针对 Amazon EC2 进行了进一步优化。 AL2023 还支持许多其他文件系统，您可以在其他卷上使用这些文件系统来满足您的特定要求。

AL2023 默认 AMIs 使用亚马逊 EBS gp3 卷，而默认使用 AL2 AMIs 亚马逊 EBS gp2 卷。您可以在启动实例时更改卷类型。

有关 Amazon EBS 卷类型的更多信息，请参阅 Amazon EBS 通用型卷。

有关启动亚马逊 EC2 实例的更多信息，请参阅亚马逊 EC2 用户指南中的启动实例。

统一控制组层次结构 (cgroup v2)

控制组 (cgroup) 是 Linux 内核的一项功能，用于分层组织进程并在进程之间分配系统资源。控制组被广泛用于通过 systemd 实现容器运行时系统。

AL2 支持cgroupv1，以及 AL2 023 支持cgroupv2。在运行容器化工作负载时（例如使用基于 AL2023 的 Amazon ECS AMI 托管容器化工作负载时），这一点值得注意。

尽管 AL2 023 仍然包含可以让系统使用运行的代码cgroupv1，但这不是推荐或支持的配置，并且将在未来的 Amazon Linux 主要版本中完全删除。

有大量关于低级 Linux 内核接口的文档以及 systemd cgroup 委派文档。

容器之外的一个常见使用案例是创建对可以使用的系统资源施加限制的 systemd 单元。有关更多信息，请参阅 systemd.resource-control。

systemd 计时器取代 cron

默认情况下，该cronie软件包安装在 AL2 AMI 上，为定期任务的传统计划crontab方式提供支持。在 AL2 023 中，cronie默认不包括在内。因此，默认不再提供对 crontab 的支持。

您可以选择安装 cronie 软件包以使用经典的 cron 作业。由于 systemd 提供附加功能，建议您迁移到 systemd 计时器。

改进的工具链：gcc、binutils 和 glibc

AL2023 包含许多与. 相同的核心软件包。 AL2

我们为 AL2 023 更新了以下三个核心工具链包。

软件包名称	AL2	AL2023
glibc	2.26	2.34
gcc	7.3	11.3
binutils	2.29	2.39

有关更多信息，请参阅 Core 工具链软件包 glibc、gcc、binutils。

有关 C、C++ 和 Fortran 语言运行时的更多信息（包括更新的默认语言标准），请参阅 AL2023 中的 C、C++ 和 Fortran。

有关优化的更多信息，请参阅性能和操作优化。

systemd 日志取代 rsyslog

在 AL2 023 中，日志系统包已更改为。 AL2 AL2默认情况下，023 不会安装，因此rsyslog默认情况下，基于文本/var/log/messages的日志文件（例如中提供的文件） AL2 不可用。 AL2023 的默认配置是systemd-journal，可以使用进行检查。journalctl尽管在 AL2 023 中rsyslog是可选的软件包，但我们建议使用systemd基于新的journalctl接口和相关软件包。有关更多信息，请参阅 journalctl 手册页面。

下表列出了一些常用 syslog 命令对应的 systmed journal 等效命令。

AL2 syslog 命令	AL2等同于 023 systemd journal
`[ec2-user ~]$ cat /var/log/messages`	`[ec2-user ~]$ journalctl`
`[ec2-user ~]$ tail -f /var/log/messages`	`[ec2-user ~]$ journalctl -f`
`[ec2-user ~]$ grep foo /var/log/messages`	`[ec2-user ~]$ journalctl | grep foo`

最小化程序包依赖

Amazon Linux 2023 对许多程序包的依赖关系图进行了最小化处理，以降低应用程序占用空间。与之相比，值得注意的变化 AL2 包括curl-minimal和gnupg-minimal软件包，它们在保留常用功能的同时，显著减少了所需软件包的数量。

主题

curl 和 libcurl 的软件包变更

AL2023 将和的常用协议和功能分离出来，libcurl打包成curlcurl-minimal和。libcurl-minimal这减少了大多数用户的磁盘、内存和依赖占用空间，并且是 AL2 023 AMIs 和容器的默认软件包。

如果需要 curl 的全部功能（例如为了获得 gopher:// 支持），请运行以下命令来安装 curl-full 和 libcurl-full 软件包。

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

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

GNU Privacy Guard (GNUPG)

AL2023 将软件包的最小和完整功能分离到gnupg2-minimal和gnupg2打包中。gnupg2-full默认情况下仅安装 gnupg2-minimal 软件包。这提供了验证 rpm 软件包上的数字签名所需的最低功能。

要从 gnupg2 获得更多功能，例如从密钥服务器下载密钥，请确保已安装 gnupg2-full 软件包。运行以下命令，将 gnupg2-minimal 交换为 gnupg2-full。

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

Amazon Corretto 作为默认 JVM

AL2023 附带亚马逊 Corretto 作为默认（也是唯一的）Java 开发套件 (JDK)。 AL2023 中所有Java基于的软件包都是使用Amazon Corretto 17构建的。

如果要从中迁移 AL2，则可以顺利地从等效OpenJDK版本过渡 AL2 到Amazon Corretto。

AWS CLIv2

AL2023 附带AWS CLI版本 2，而 AL2 随版本 1 一起提供。AWS CLI

UEFI 首选和安全启动

默认情况下，在支持 UEFI 固件的实例类型上使用 AL2 023 AMI 启动的任何实例都将在 UEFI 模式下启动。这是通过将启动模式 AMI 参数设置为 `uefi-preferred` 来完成的。有关更多信息，请参阅 Amazon EC2 用户指南中的启动模式。

在支持 UEFI 安全启动的亚马逊 EC2 实例类型上，可以在亚马逊 Linux 2023 中启用安全启动。有关更多信息，请参阅 AL2023 上的 UEFI 安全启动。

SSH 服务器默认配置更改

对于 AL2 023 AMI，我们更改了随版本生成的`sshd`主机密钥的类型。我们还删除了一些旧密钥类型，以避免在启动时生成它们。客户端必须通过使用 `ed25519` 密钥支持 `rsa-sha2-256` 和 `rsa-sha2-512` 协议或 `ssh-ed25519`。默认情况下，`ssh-rsa` 签名处于禁用状态。

此外，默认`sshd_config`文件中的 AL2 023 配置设置包含`UseDNS=no`。这一新设置意味着 DNS 受损不太可能阻碍您与实例建立 `ssh` 会话。但代价在于，`authorized_keys` 文件中的 `from=`*`hostname.domain`*`,`*`hostname.domain`* 行条目将无法解析。由于 `sshd` 不再尝试解析 DNS 名称，因此必须将每个以逗号分隔的 *`hostname.domain`* 值转换为对应的 IP address。

有关更多信息，请参阅 默认 SSH 服务器配置。

AL2023 内核更改自 AL2

AL2023 带来了 6.1 内核以及许多配置更改，以进一步优化 Amazon Linux 的云端应用。对于大多数用户而言，这些变化应该是完全透明的。

IPv4 TTL

的 TTL IPv4 是通过配置的`sysctl`，默认值显示在中。`/etc/sysctl.d/00-defaults.conf`此值可通过通常的 `sysctl` 方法进行自定义。有关更多信息，请参阅 `sysctl` man 页面。

AL2 将该net.ipv4.ip_default_ttl值设置为 255，同时将其 AL2023 设置为 127。这使得 Amazon Linux 的默认值与其他主流 Linux 发行版保持一致。若无明确需求，不建议更改此默认值。

注重安全的内核配置变化

CONFIG 选项	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 G_ON_DA _CORRUP ON	n	y	n	y	y	y	y	y
CONFIG_I FAULT_MI P_MIN_AI R	4096	4096	4096	4096	65536	65536	65536	65536
CONFIG_I VMEM	n	y	n	y	n	n	n	n
CONFIG_I VPORT	n	y	n	y	n	n	n	n
CONFIG_I RTIFY_SC RCE	n	y	n	y	y	y	y	y
CONFIG_I RDENED_U ERCOPY_I LLBACK	不适用	不适用	y	y	不适用	不适用	不适用	不适用
CONFIG_ IT_ON_AI OC_DEFAU T_ON	不适用	不适用	n	n	n	n	n	n

CONFIG 选项	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_ IT_ON_FI E_DEFAU _ON	不适用	不适用	n	n	n	n	n	n
CONFIG_ MMU_DEF LT_DMA_ RICT	不适用	不适用	不适用	不适用	n	n	n	n
CONFIG_ ISC_AUT OAD	y	y	y	y	n	n	n	n
CONFIG_ HED_COR	不适用	不适用	不适用	不适用	不适用	y	不适用	y
CONFIG_ HED_STA _END_CH K	n	y	n	y	y	y	y	y
CONFIG_ CURITY_ ESG_RES ICT	n	n	n	n	y	y	y	y
CONFIG_ CURITY_ LINUX_D ABLE	y	y	y	y	n	n	不适用	不适用

CONFIG 选项	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_S UFFLE_PA E_ALLOCA OR	不适用	不适用	y	y	y	y	y	y
CONFIG_S AB_FREEL ST_HARDE ED	n	y	y	y	y	y	y	y
CONFIG_S AB_FREEL ST_RANDO	n	n	y	y	y	y	y	y

x86-64 特定安全相关内核配置变更

CONFIG 选项	AL2/4.14/x86_64	AL2/5.10/x86_64	AL2023/6.1/ x86_64	AL2023/6.12/ x86_64
CONFIG_AM D_IOMMU	y	y	y	y
CONFIG_AM D_IOMMU_V2	m	m	y	不适用
CONFIG_RA NDOMIZE_M EMORY	不适用	y	y	y

aarch64（ARM/Graviton）特定安全相关内核配置变更

CONFIG 选项	AL2/4.14/aarch64	AL2/5.10/aarch64	AL2023/6.1/aarch64	AL2023/6.12/aarch64
CONFIG_ARM64_PTR_AUTH	不适用	y	y	y
CONFIG_ARM64_PTR_AUTH_KERNEL	不适用	不适用	y	y
CONFIG_ARM64_SW_TTBR0_PAN	y	y	y	y

/dev/mem、**/dev/kmem** 和 **/dev/port**

Amazon Linux 2023 在已有的限制的基础上完全禁用/dev/memCONFIG_DEVMEM和/dev/port（而且CONFIG_DEVPORT）。 AL2

在 5.13 内核中，该/dev/kmem代码已从 Linux 中完全删除，虽然它在中被禁用 AL2，但现在不适用于 AL2023。

此选项是内核自我保护项目推荐设置之一。

FORTIFY_SOURCE

AL2023 CONFIG_FORTIFY_SOURCE在所有支持的架构上启用。此功能是一项安全强化功能。在编译器可以确定和验证缓冲区大小的情况下，此功能可以检测常见字符串和内存函数中的缓冲区溢出。

此选项是内核自我保护项目推荐设置之一。

行规程自动加载（**CONFIG_LDISC_AUTOLOAD**）

除非请求来自具有CAP_SYS_MODULE权限的进程，否则 AL2023 内核不会自动加载线路规范TIOCSETDioctl，例如通过使用中的软件加载。

此选项是内核自我保护项目推荐设置之一。

非特权用户对 **dmesg** 的访问 (**CONFIG_SECURITY_DMESG_RESTRICT**)

默认情况下，AL2023 不允许非特权用户访问。dmesg

此选项是内核自我保护项目推荐设置之一。

SELinux **selinuxfs**禁用

AL2023 禁用已弃用的CONFIG_SECURITY_SELINUX_DISABLE内核选项，该选项启用了一种在加载策略 SELinux 之前禁用的运行时方法。

此选项是内核自我保护项目推荐设置之一。

其他内核配置变化

CONFIG 选项	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 _CPUS	4096	8192	4096	8192	4096	8192	4096	8192
CONFIG_I NIC_ON_(PS	y	n	y	n	y	y	y	y
CONFIG_I NIC_ON_(PS_VALUI	1	0	1	0	1	1	1	1
CONFIG_I P	m	m	m	m	n	n	n	n
CONFIG_S IP	m	m	m	m	n	n	n	n

CONFIG 选项	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_ N_PV	不适用	y	不适用	n	不适用	n	不适用	n

CONFIG_HZ

AL2023 CONFIG_HZ在两个aarch64平台上都x86-64设置为 100。

CONFIG_NR_CPUS

AL2023 设置CONFIG_NR_CPUS为一个接近 Amazon EC2 中最大 CPU 内核数的数字。

内核错误处理

内 AL2023 核在运行时会死机。此功能等同于在内核命令行上使用 oops=panic 引导。

内核错误是指内核检测到可能影响系统的进一步可靠性的内部错误。

PPP 和 SLIP 支持

AL2023 不支持 PPP 或 SLIP 协议。

Xen PV 访客支持

AL2023 不支持以 Xen PV 访客身份运行。

内核文件系统支持

内核支持挂载的文件系统发生了几处变化，内核 AL2 将要解析的分区方案也发生了变化。

CONFIG 选项	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_ S_FS	n	m	n	m	n	n	n	n

CONFIG 选项	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_ _RXRPC_	n	m	n	m	n	n	n	n
CONFIG_ D_DISKL EL_	y	y	y	y	n	n	n	n
CONFIG_ AMFS_	m	m	m	m	n	n	n	n
CONFIG_ AMFS_BL KDEV_	不适用	不适用	y	n	不适用	不适用	不适用	不适用
CONFIG_ _CLONE_	不适用	不适用	n	n	n	n	n	n
CONFIG_ _ERA_	m	n	m	n	n	n	n	n
CONFIG_ _INTEGR Y_	n	m	n	m	m	m	m	m
CONFIG_ _LOG_WR ES_	n	n	m	m	m	m	m	m
CONFIG_ _SWITCH	m	n	m	n	n	n	n	n
CONFIG_ _VERITY	m	n	m	n	m	m	m	m

CONFIG 选项	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 RYPT_FS	n	m	n	m	n	n	n	n
CONFIG_I FAT_FS	不适用	不适用	m	m	m	m	m	m
CONFIG_I T2_FS	n	m	n	m	n	n	n	n
CONFIG_I T3_FS	n	m	n	m	n	n	n	n
CONFIG_(S2_FS	m	m	m	m	n	n	n	n
CONFIG_I SPLUS_F	n	m	n	m	n	n	n	n
CONFIG_I S_FS	n	m	n	m	n	n	n	n
CONFIG_ S_FS	n	n	n	n	n	n	n	n
CONFIG_I M_PARTI ON	n	y	n	y	n	n	n	n
CONFIG_I C_PARTI ON	n	y	n	y	n	n	n	n
CONFIG_I S_V2	n	m	n	m	n	n	n	n

CONFIG 选项	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 FS_FS	n	m	n	n	n	n	n	n
CONFIG_I MFS_FS	n	m	n	m	n	n	n	n
CONFIG_S LARIS_X8 _PARTIT N	n	y	n	y	n	n	n	n
CONFIG_S UASHFS_Z TD	n	y	n	y	y	y	y	y
CONFIG_S N_PARTI ON	n	y	n	y	n	n	n	n

Andrew 文件系统支持 (AFS)

内核不再支持afs文件系统。 AL2 未附带的用户空间支持。afs

cramfs 支持

内核不再支持 cramfs 文件系统。中的继任者 AL2023 是squashfs文件系统。

支持 BSD 磁盘标签

内核不再支持 BSD 磁盘标签。如果需要读取带有 BSD 磁盘标签的卷，则 BSDs可以启动各种卷标。

设备映射器更改

对 AL2023 内核中配置的设备映射器目标进行了几处更改。

eCryptFs 支持

Amazon Linux 中已弃用了 ecryptfs 文件系统。的用户空间组件ecryptfs已存在于 AL1、已移除中 AL2，并且 AL2023 不再在ecryptfs支持下构建内核。

exFAT

在 5.10 内核中 AL2添加了对exFAT文件系统的支持。它在 4.14 内核 AL2 发布时不存在。 AL2023 继续支持exFAT文件系统。

ext2、ext3 和 ext4 文件系统

AL2023 随附CONFIG_EXT4_USE_FOR_EXT2选项，这意味着ext4文件系统代码将用于读取传统ext2文件系统。

CONFIG_ _FS GFS2

该内核不再是使用 CONFIG_ GFS2 _FS 构建的。

Apple Extended HFS 文件系统支持 (HFS+)

在中 AL2，只有内x86-64核是在支持hfsplus文件系统的情况下构建的。 AL2 5.15 内核不hfsplus支持任何架构。在中 AL2023，我们完成了对亚马逊 Linux hfsplus 支持的弃用。

HFS 文件系统支持

在中 AL2，只有内x86-64核是在支持hfs文件系统的情况下构建的。 AL2 5.15 内核不hfs支持任何架构。在中 AL2023，我们完成了对亚马逊 Linux hfs 支持的弃用。

JFS 文件系统支持

较旧的 AL2 x86-64内核是在支持jfs文件系统的情况下构建的。 AL2 5.15 内核不jfs支持任何架构。两者都不是， AL1 AL2 或者与 JFS 用户空间一起提供。在中 AL2023，我们完成了对亚马逊 Linux jfs 支持的弃用。

上游 Linux 内核正在考虑移除 JFS。因此，如果您在 JFS 文件系统上存有数据，应将其迁移到其他文件系统。2024 年，JFS 已从所有当前版本的 Amazon Linux 内核中移除。

Windows 逻辑磁盘管理器（动态磁盘）支持（**CONFIG_LDM_PARTITION**）

AL2023 不再支持Windows 2000Windows XP、或带有MS-DOS样式分区的Windows Vista动态磁盘。此代码从来不支持 Windows Vista 所引入的基于 GPT 的较新的动态磁盘。

Macintosh 分区映射支持

AL2023 不再支持经典的 Macintosh 分区图。默认情况下，现代 macOS 版本会创建现代 GPT 分区表，以取代这种较旧的类型。

NFSv2 支持

AL2023 不再支持 NFSv2，但继续支持 NFSv3、 NFSv4、 NFSv4 .1 和 NFSv4 .2。我们建议您迁移到 NFSv3 或更新版本。

NTFS (**CONFIG_NTFS_FS**)

从5.10内核`ntfs`开始，在亚马逊 Linux 上访问 NTFS 文件系统的`ntfs3`代码已被替换。 AL2 AL2023 不再包含该`ntfs`代码，而是完全依赖该`ntfs3`代码来访问 NTFS 文件系统。

romfs 文件系统

在 `squashfs` Amazon Linux 中，`romfs`文件系统是文件系统的继任者，并且该 AL2023 内核的构建不再支持了`romfs`。

Solaris x86 硬盘分区格式

AL2023 不再支持 Solaris x86 硬盘分区格式。

squashfszstd压缩

AL2023 在所有支持的架构上添加了对zstd压缩`squashfs`文件系统的支持。

Sun 分区表支持

AL2023 不再支持 Sun 分区表格式 (`CONFIG_SUN_PARTITION`)。

/tmp 现在是 tmpfs

与 Amazon Linux 2 相比，Amazon Linux 2023 引入了 `/tmp` 行为方式的更改。的默认配置 AL2 是，`/tmp`和`/var/tmp`都在根文件系统上。Amazon Linux 2023 默认使用 `tmpfs` 作为 `/tmp`，限制为 RAM 的 50% 且最多一百万个 inodes。这些更改使 Amazon Linux 的行为与其他 Linux 发行版保持一致。

有关 AL2 023 文件系统布局的完整详细信息，请参阅文件系统布局部分`/var/tmp`中的`/tmp`和。

AMI 和容器映像变化

AMIs 和容器中包含的包裹已发生一些变化。

Amazon Linux 2023 引入了the section called “AL2023 最小容器镜像”，并支持构建the section called “构建基本容器镜像 AL2023”。有关更多信息，请参阅 AL2023 在容器中使用。

比较 Amazon Linux 2 和 Amazon Linux 2023 AMI 上安装的软件包

Amazon Linux 2 与 AL2023 标准 AMI 中存在的 RPM 程序包比较。

软件包	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

软件包	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	2.1.1
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

软件包	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

软件包	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	

软件包	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

软件包	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

软件包	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

软件包	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-langpack	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）

软件包	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

软件包	AL2 AMI	AL2023 AMI
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	2.10	2.14
jbigkit-libs	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-repo-s3		2023.6.20241031
kernel-srpm-macros		1.0
kernel-tools	5.10.228	6.1.112

软件包	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

软件包	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 (x8_64)	
libeconf		0.4.0
libedit	3.0	3.1
libestr	0.1.9	

软件包	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-hmaccalc		1.4.0
libldb		2.6.2
libmaxminddb		1.5.2

软件包	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_conntrack	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

软件包	AL2 AMI	AL2023 AMI
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
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	

软件包	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

软件包	AL2 AMI	AL2023 AMI
libxml2-python	2.9.1	
libyaml	0.1.4	0.2.5
libzstd		1.5.5
linux-firmware-whence		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	

软件包	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

软件包	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

软件包	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

软件包	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

软件包	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

软件包	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	
policycoreutils	2.5	3.4
policycoreutils-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	

软件包	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

软件包	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

软件包	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

软件包	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	

软件包	AL2 AMI	AL2023 AMI
python-backports	1.0	
python-backports-ssl_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	

软件包	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

软件包	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
setuptool	1.19.11	

软件包	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	

软件包	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

软件包	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	flink-client	

软件包	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

比较 Amazon Linux 2 和 Amazon Linux 2023 Minimal AMI 上安装的软件包

Amazon Linux 2 与 AL2023 AMI 最低版本中存在的 RPM 程序包比较。

软件包	AL2 最低版本	AL2023 最低版本
acl	2.2.51	
alternatives		1.15
amazon-chrony-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

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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-langpack	2.26	

软件包	AL2 最低版本	AL2023 最低版本
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	

软件包	AL2 最低版本	AL2023 最低版本
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-repo-s3		2023.6.20241031
keyutils-libs	1.5.8	1.6.3

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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-hmaccalc		1.4.0
libmetalink	0.1.3	

软件包	AL2 最低版本	AL2023 最低版本
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	
libnghttp2	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

软件包	AL2 最低版本	AL2023 最低版本
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-whence		20210208（noarch）
logrotate	3.8.6	3.20.1
lua	5.1.4	
lua-libs		5.4.4

软件包	AL2 最低版本	AL2023 最低版本
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	

软件包	AL2 最低版本	AL2023 最低版本
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	

软件包	AL2 最低版本	AL2023 最低版本
pcre2	10.23	10.40
pcre2-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	

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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-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
python3-ply		3.11
python3-policycoreutils		3.4
python3-prettytable		0.7.2
python3-prompt-toolkit		3.0.24
python3-pycparser		2.20

软件包	AL2 最低版本	AL2023 最低版本
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	

软件包	AL2 最低版本	AL2023 最低版本
python-backports-ssl_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	

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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

软件包	AL2 最低版本	AL2023 最低版本
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
xfsprogs	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	flink-client	
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

比较 Amazon Linux 2 和 Amazon Linux 2023 基础容器映像上安装的软件包

Amazon Linux 2 与 AL2023 基础容器映像中存在的 RPM 程序包比较。

软件包	AL2 容器	AL2023 容器
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	

软件包	AL2 容器	AL2023 容器
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

软件包	AL2 容器	AL2023 容器
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

软件包	AL2 容器	AL2023 容器
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-filesystem		2.15.2
libselinux	2.5	3.4
libsepol	2.5	3.4
libsigsegv		2.13
libsmartcols		2.37.4

软件包	AL2 容器	AL2023 容器
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	

软件包	AL2 容器	AL2023 容器
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-libs	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	

软件包	AL2 容器	AL2023 容器
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

软件包	AL2 容器	AL2023 容器
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	flink-client	
yum-plugin-ovl	1.1.31	
yum-plugin-priorities	1.1.31	
zlib	1.2.7	1.2.11

比较 AL1 和 AL2023

以下主题描述了 AL1 与 AL2023 之间未在与 AL2 的比较中涵盖的关键差异。

> **Note**
>
> AL1 于 2023 年 12 月 31 日结束其生命周期（EOL），并将自 2024 年 1 月 1 日起停止接收任何安全更新或错误修复。有关 AL1 EOL 和维护支持的更多信息，请参阅博客文章关于 Amazon Linux AMI 生命周期终止的最新动态。我们建议您将应用程序升级到 AL2023，其中包含直到 2028 年的长期支持。

主题

针对每个版本的支持

对于 AL2023，我们自发布之日起提供五年支持。AL1 已于 2020 年 12 月 31 日终止标准支持，并于 2023 年 12 月 31 日终止维护支持。

有关更多信息，请参阅 发布频率。

`systemd` 取代 `upstart` 作为 `init` 系统

在 AL2 中，`systemd` 取代 `upstart` 作为 `init` 系统。AL2023 也使用 `systemd` 作为其 `init` 系统，并且进一步采用了 `systemd` 的新特性和功能。

Python 2.6 和 2.7 已被 Python 3 所取代

尽管 AL1 在 2018.03 版本中已将 Python 2.6 标记为 EOL，但这些程序包在存储库中仍可安装。AL2 最初支持的 Python 版本为 Python 2.7，而 AL2023 完成了向 Python 3 的过渡。AL2023 存储库中不包含任何 Python 2.x 版本。

有关 Amazon Linux 上的 Python 的更多信息，请参阅 AL2023 中的 Python。

OpenJDK 8 是最旧的 JDK

AL2023 附带 Amazon Corretto 作为默认（也是唯一的）Java 开发工具包 (JDK)。AL2023 中所有基于 Java 的软件包均使用 Amazon Corretto 17 构建。

在 AL1 中，OpenJDK 1.6.0 (`java-1.6.0-openjdk`) 于第一个 2018.03 版本中 EOL，OpenJDK 1.7.0 (`java-1.7.0-openjdk`) 于 2020 中期 EOL，不过这两种版本在 AL1 存储库中都可用。AL2023 中可用的最早的 OpenJDK 版本是 OpenJDK 8，由 Amazon Corretto 8 提供。

AL2亚马逊 Linux 上的 023 个内核更改 1 () AL1

内核实时修补

023 和 AL2 023 都 AL2 增加了对内核实时补丁功能的支持。这使您无需重启或停机即可修补 Linux 内核中的关键和重要安全漏洞。有关更多信息，请参阅 AL2023 上的内核实时修补。

内核文件系统支持

内核 AL1 将支持挂载的文件系统发生了几处变化，内核将要解析的分区方案也发生了变化。

CONFIG 选项	AL1/4.14/ x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_AFS_FS	m	n	n	n	n
CONFIG_AF_RXRPC	m	n	n	n	n

CONFIG 选项	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_BSD_DISKLABEL	y	n	n	n	n
CONFIG_CRAMFS	m	n	n	n	n
CONFIG_CRAMFS_BLOCKDEV	不适用	不适用	不适用	不适用	不适用
CONFIG_DM_CLONE	不适用	n	n	n	n
CONFIG_DM_ERA	n	n	n	n	n
CONFIG_DM_INTEGRITY	m	m	m	m	m
CONFIG_DM_LOG_WRITES	n	m	m	m	m
CONFIG_DM_SWITCH	n	n	n	n	n
CONFIG_DM_VERITY	n	m	m	m	m
CONFIG_ECRYPT_FS	m	n	n	n	n
CONFIG_EXFAT_FS	不适用	m	m	m	m

CONFIG 选项	AL1/4.14/ x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_EXT2_FS	m	n	n	n	n
CONFIG_EXT3_FS	m	n	n	n	n
CONFIG_GFS2_FS	n	n	n	n	n
CONFIG_HFSPLUS_FS	m	n	n	n	n
CONFIG_HFS_FS	m	n	n	n	n
CONFIG_JFS_FS	n	n	n	n	n
CONFIG_LDM_PARTITION	y	n	n	n	n
CONFIG_MAC_PARTITION	y	n	n	n	n
CONFIG_NFS_V2	m	n	n	n	n
CONFIG_NTFS_FS	m	n	n	n	n
CONFIG_ROMFS_FS	m	n	n	n	n

CONFIG 选项	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
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

注重安全的内核配置变化

CONFIG 选项	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_DEVMEM	y	n	n	n	n
CONFIG_DEVPORT	y	n	n	n	n

CONFIG 选项	AL1/4.14/ x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_FORTIFY_SOURCE	y	y	y	y	y
CONFIG_HARDENED_USERCOPY_FALLBACK	不适用	不适用	不适用	不适用	不适用
CONFIG_INIT_ON_ALLOC_DEFAULT_ON	不适用	n	n	n	n
CONFIG_INIT_ON_FREE_DEFAULT_ON	不适用	n	n	n	n
CONFIG_IOMMU_DEFAULT_DMA_STRICT	不适用	n	n	n	n
CONFIG_LDISC_AUTOLOAD	y	n	n	n	n
CONFIG_SCHED_CORE	不适用	不适用	y	不适用	y
CONFIG_SCHED_STACK_END_CHECK	y	y	y	y	y

CONFIG 选项	AL1/4.14/ x86_64	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_SECURITY_DMESG_RESTRICT	n	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	y	n	n	不适用	不适用
CONFIG_SHUFFLE_PAGE_ALLOCATOR	不适用	y	y	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y	y	y	y
CONFIG_SLAB_FREELIST_RANDOM	n	y	y	y	y

其他内核配置变化

CONFIG 选项	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 选项	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_PANIC_ON_OOPS	n	y	y	y	y
CONFIG_PANIC_ON_OOPS_VALUE	0	1	1	1	1
CONFIG_PPP	m	n	n	n	n
CONFIG_SLIP	m	n	n	n	n
CONFIG_XEN_PV	y	不适用	n	不适用	n

比较 Amazon Linux 1 (AL1) 和 Amazon Linux 2023 AMI 上安装的软件包

AL1 与 AL2023 标准 AMI 中存在的 RPM 程序包比较。

软件包	AL1 AMI	AL2023 AMI
acl	2.2.49	2.3.1
acpid	2.0.19	2.0.32
alsa-lib	1.0.22	
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1

软件包	AL1 AMI	AL2023 AMI
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
bash-completion		2.1.1
bc	1.06.95	1.07.1
bind-libs	9.8.2	9.18.28
bind-license		9.18.28

软件包	AL1 AMI	AL2023 AMI
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
copy-jdk-configs	3.3	
coreutils	8.22	8.32
coreutils-common		8.32
cpio	2.10	2.13

软件包	AL1 AMI	AL2023 AMI
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	
db4-utils	4.7.25	
dbus	1.6.12	1.12.28
dbus-broker		32
dbus-common		1.12.28

软件包	AL1 AMI	AL2023 AMI
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
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0

软件包	AL1 AMI	AL2023 AMI
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
ec2-instance-connect-selinux		1.1
ec2-net-utils	0.7	
ec2-utils	0.7	2.2.0
ed	1.1	1.14.2

软件包	AL1 AMI	AL2023 AMI
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
findutils	4.4.2	4.8.0
fipscheck	1.3.1	
fipscheck-lib	1.3.1	
fontconfig	2.8.0	

软件包	AL1 AMI	AL2023 AMI
fontpackages-filesystem	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
glibc	2.17	2.34
glibc-all-langpacks		2.34
glibc-common	2.17	2.34
glibc-gconv-extra		2.34

软件包	AL1 AMI	AL2023 AMI
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
grub2-tools-minimal		2.06
grubby	7.0.15	8.40
gssproxy		0.8.4
gzip	1.5	1.12

软件包	AL1 AMI	AL2023 AMI
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
irqbalance	1.5.0	1.9.0
jansson		2.14
java-1.7.0-openjdk	1.7.0.321	
javapackages-tools	0.9.1	

软件包	AL1 AMI	AL2023 AMI
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-repo-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
kpartx	0.4.9	
kpatch-runtime		0.9.7
krb5-libs	1.15.1	1.21.3
lcms2	2.6	

软件包	AL1 AMI	AL2023 AMI
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
libcgroup	0.40.rc1	
libcollection		0.7.0
libcom_err	1.43.5	1.46.5
libcomps		0.1.20
libconfig		1.7.2
libcurl	7.61.1	
libcurl-minimal		8.5.0

软件包	AL1 AMI	AL2023 AMI
libdb		5.3.28
libdhash		0.5.0
libdnf		0.69.0
libeconf		0.4.0
libedit	2.1.1	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
libgssglue	0.1	
libibverbs		48.0
libICE	1.0.6	
libicu	50.2	

软件包	AL1 AMI	AL2023 AMI
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_conntrack	1.0.4	
libnfnetlink	1.0.1	
libnfsidmap	0.25	2.5.4
libnghttp2	1.33.0	1.59.0
libnih	1.0.1	
libnl	flink-client	
libnl3		3.5.0
libpath_utils		0.2.1

软件包	AL1 AMI	AL2023 AMI
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	
libsmartcols	2.23.2	2.37.4
libsolv		0.7.22
libss	1.43.5	1.46.5
libssh2	1.4.2	

软件包	AL1 AMI	AL2023 AMI
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
libuser	0.60	0.63
libutempter	1.1.5	1.2.1
libuuid	2.23.2	2.37.4
libuv		1.47.0

软件包	AL1 AMI	AL2023 AMI
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	
libyaml	0.1.6	0.2.5
libzstd		1.5.5
linux-firmware-whence		20210208
lm_sensors-libs		3.6.0

软件包	AL1 AMI	AL2023 AMI
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	
microcode_ctl	2.1	2.1
mingetty	1.08	
mpfr		4.1.0
nano	2.5.3	5.8
nc	1.84	

软件包	AL1 AMI	AL2023 AMI
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
nss-tools	3.53.1	
nss-util	3.53.1	3.90.0
ntp	4.2.8p15	
ntpdate	4.2.8p15	
ntsysv	1.3.49.3	1.15

软件包	AL1 AMI	AL2023 AMI
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-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	

软件包	AL1 AMI	AL2023 AMI
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	
perl-DynaLoader		1.47
perl-Encode	2.51	3.15
perl-Errno		1.30
perl-Exporter	5.68	5.74
perl-Fcntl		1.13

软件包	AL1 AMI	AL2023 AMI
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
perl-mro		1.23
perl-overload		1.31
perl-overloading		0.02
perl-parent	0.225	0.238
perl-PathTools	3.40	3.78

软件包	AL1 AMI	AL2023 AMI
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
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	

软件包	AL1 AMI	AL2023 AMI
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
psacct	6.3.2	6.6.4
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa		20240212

软件包	AL1 AMI	AL2023 AMI
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	
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	

软件包	AL1 AMI	AL2023 AMI
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-pygpgme	0.3	
python27-pyliblzma	0.5.3	
python27-pystache	0.5.3	
python27-pyxattr	0.5.0	
python27-PyYAML	3.10	
python27-requests	1.2.3	
python27-rsa	3.4.1	
python27-setuptools	36.2.7	
python27-simplejson	3.6.5	

软件包	AL1 AMI	AL2023 AMI
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
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

软件包	AL1 AMI	AL2023 AMI
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-jsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-libselinux		3.4
python3-libsemanage		3.4
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

软件包	AL1 AMI	AL2023 AMI
python3-ply		3.11
python3-policycoreutils		3.4
python3-prettytable		0.7.2
python3-prompt-toolkit		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

软件包	AL1 AMI	AL2023 AMI
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
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

软件包	AL1 AMI	AL2023 AMI
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
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	

软件包	AL1 AMI	AL2023 AMI
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	
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

软件包	AL1 AMI	AL2023 AMI
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	
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

软件包	AL1 AMI	AL2023 AMI
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
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

软件包	AL1 AMI	AL2023 AMI
yum-metadata-parser	flink-client	
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

比较 Amazon Linux 1 (AL1) 和 Amazon Linux 2023 Minimal AMI 上安装的软件包

AL1 与 AL2023 AMI 最低版本中存在的 RPM 程序包比较。

软件包	AL1 最低版本	AL2023 最低版本
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

软件包	AL1 最低版本	AL2023 最低版本
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
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

软件包	AL1 最低版本	AL2023 最低版本
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	
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

软件包	AL1 最低版本	AL2023 最低版本
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
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	

软件包	AL1 最低版本	AL2023 最低版本
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
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	

软件包	AL1 最低版本	AL2023 最低版本
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
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

软件包	AL1 最低版本	AL2023 最低版本
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
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

软件包	AL1 最低版本	AL2023 最低版本
kbd-misc	1.15	2.4.0
kernel	4.14.336	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-repo-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
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	

软件包	AL1 最低版本	AL2023 最低版本
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	2.1.1	3.1
libfdisk		2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0
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

软件包	AL1 最低版本	AL2023 最低版本
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
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
libsmartcols	2.23.2	2.37.4

软件包	AL1 最低版本	AL2023 最低版本
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
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-whence		20210208
logrotate	3.7.8	3.20.1

软件包	AL1 最低版本	AL2023 最低版本
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
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	

软件包	AL1 最低版本	AL2023 最低版本
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	
ntpdate	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
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
pam	1.1.8	1.5.1
passwd	0.79	0.80

软件包	AL1 最低版本	AL2023 最低版本
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
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	
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	

软件包	AL1 最低版本	AL2023 最低版本
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-pygpgme	0.3	
python27-pyliblzma	0.5.3	
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

软件包	AL1 最低版本	AL2023 最低版本
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
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21

软件包	AL1 最低版本	AL2023 最低版本
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-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
python3-ply		3.11
python3-policycoreutils		3.4
python3-prettytable		0.7.2
python3-prompt-toolkit		3.0.24
python3-pycparser		2.20
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1

软件包	AL1 最低版本	AL2023 最低版本
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
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

软件包	AL1 最低版本	AL2023 最低版本
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	
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

软件包	AL1 最低版本	AL2023 最低版本
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	
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

软件包	AL1 最低版本	AL2023 最低版本
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	flink-client	
yum-plugin-priorities	1.1.31	
yum-plugin-upgrade-helper	1.1.31	
zlib	1.2.8	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2
zstd		1.5.5

比较 Amazon Linux 1 (AL1) 和 Amazon Linux 2023 基础容器映像上安装的软件包

AL1 与 AL2023 基础容器映像中存在的 RPM 程序包比较。

软件包	AL1 容器	AL2023 容器
alternatives		1.15

软件包	AL1 容器	AL2023 容器
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
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-yama-scope		0.188
elfutils-libelf	0.168	0.188

软件包	AL1 容器	AL2023 容器
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
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

软件包	AL1 容器	AL2023 容器
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
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	

软件包	AL1 容器	AL2023 容器
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-filesystem		2.15.2
libselinux	2.1.10	3.4
libsepol	2.1.7	3.4
libsigsegv		2.13
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

软件包	AL1 容器	AL2023 容器
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	
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	

软件包	AL1 容器	AL2023 容器
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	
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	

软件包	AL1 容器	AL2023 容器
python27-pygpgme	0.3	
python27-pyliblzma	0.5.3	
python27-pyxattr	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
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

软件包	AL1 容器	AL2023 容器
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
yum	3.4.3	4.14.0
yum-metadata-parser	flink-client	
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 系统要求

这部分描述使用 AL2023 的系统要求。

主题

运行 AL2023 的 CPU 要求

要运行任何 AL2023 代码，所使用的处理器需要满足某些最低要求。尝试在不满足这些要求的 CPU 上运行 AL2023 可能会导致在代码执行初期就出现非法指令错误。

最低要求适用于 Amazon EC2 上的 AL2023、AL2023 在容器里 和 Amazon EC2 之外的 AL2023。

AL2023 的 ARM CPU 要求

所有 AL2023 `aarch64` (ARM) 二进制文件均为 64 位构建。不提供 32 位 ARM 二进制文件，因此需要 64 位 ARM CPU。

> **Note**
>
> 对于基于 ARM 的实例，AL2023 仅支持使用 Graviton2 或更高版本处理器的实例类型。AL2023 不支持 A1 实例。

AL2023 需要带有加密扩展 (`ARMv8.2+crypto`) 的 ARMv8.2 兼容处理器。所有用于 `aarch64` 的 AL2023 程序包均使用 `-march=armv8.2-a+crypto` 编译器标志构建。尽管我们尝试在较旧的 ARM 处理器上运行 AL2023 代码时显示友好的错误消息，但第一条错误消息仍可能是非法指令错误。

> **Note**
>
> 由于 AL2023 `aarch64` 的基础 CPU 要求，所有在 Raspberry Pi 5 之前的 Raspberry Pi 系统均不满足最低 CPU 要求。

AL2023 的 x86-64 CPU 要求

所有 AL2023 x86-64 二进制文件均通过向编译器传递 `-march=x86-64-v2` 为 x86-64 架构的 x86-64v2 修订版构建。

该架构的 x86-64v2 修订版在基线 x86-64 架构之上增加了以下 CPU 特性：

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

这大致对应于 2009 年或之后发布的 x86-64 处理器。示例包括 Intel Nehalem、AMD Jaguar、Atom Silvermont，以及 VIA Nano 和 Eden C 微架构。

在 Amazon EC2 中，所有 x86-64 实例类型都支持 x86-64v2，包括 M1、C1 和 M2 实例系列。

未构建 32 位 x86 (i686) AL2023 二进制文件。尽管 AL2023 保留了对运行 32 位用户空间二进制文件的支持，但此功能已弃用，并可能在未来的 Amazon Linux 主要版本中移除。有关更多信息，请参阅 32 位 x86 (i686) 软件包。

运行 AL2023 的内存（RAM）要求

Amazon EC2 .nano 系列实例类型（t2.nano、t3.nano、t3a.nano 和 t4g.nano）具有 512 MB RAM，这是 AL2023 的最低要求。

> **Note**
>
> 虽然 512 MB 是最低要求，但这些实例类型内存受限，功能和性能可能受限。

AL2023 映像未在内存少于 512 MB 的系统上测试。在少于 512 MB RAM 的环境中运行基于 AL2023 的容器映像将取决于容器化工作负载。

某些工作负载，例如某些 AL2023 发布版本间的 `dnf upgrade`，可能需要超过 512 MB RAM。因此，AL2023.3 发布版本默认对内存少于 800 MB 的实例启用了 `zram`。对于容器化工作负载，这意味着某些工作负载在具有此内存量的 AL2023 实例上可能运行良好，但在限制为此内存使用量的容器中运行时可能会失败。

对于 RAM 小于 800MB 的实例类型，AL2023（从 AL2023.3 开始）将默认启用基于 `zram` 的交换。内存少于 800 MB 的 Amazon EC2 实例类型示例包括 `t4g.nano`、`t3a.nano`、`t3.nano`、`t2.nano` 和 `t1.micro`。这意味着这些实例类型的内存不足情况更少，因为 AL2023 将按需压缩和解压缩内存页面。这可以支持原本需要具有更多内存的实例类型的工作负载，但会以增加压缩所需的 CPU 使用量为代价。

亚马逊 Linux 内核

根据上游 Linux 社区的年度 LTS 内核，亚马逊 Linux 2023 (AL2023) 在每年第一季度发布新的长期支持 (LTS) 内核。每个新的 LTS 内核都带来了上游 Linux 内核的最新安全补丁、性能改进、硬件支持和功能。

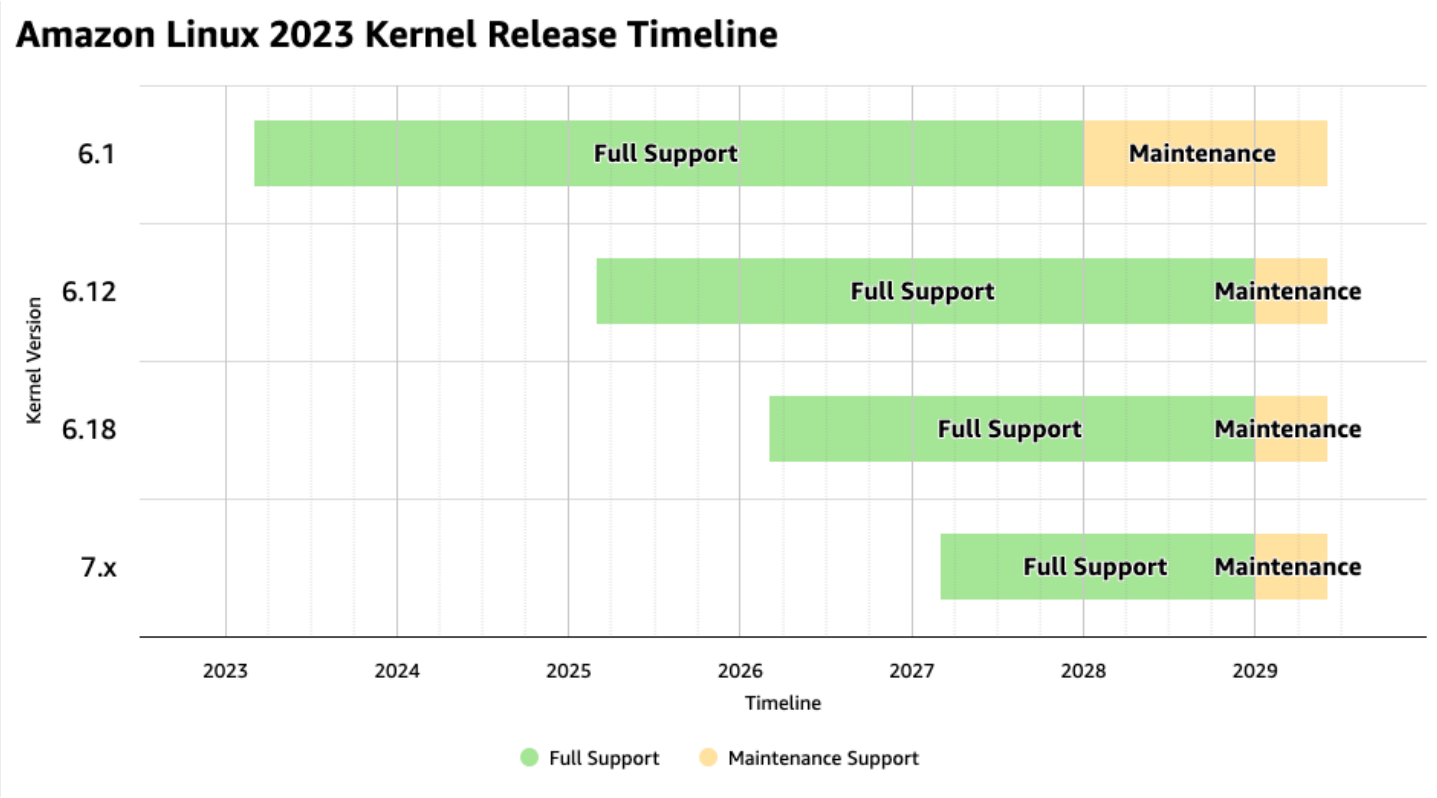
内核生命周期

每个 AL2023 LTS 内核的支持期限为四年，分为两个阶段：

1. 全面支持（第 1-2 年）— 内核通过定期在上游 LTS 内核上进行变基，获得所有 CVE 严重程度的修复。此阶段与上游 Linux 社区的 LTS 支持窗口保持一致。如果上游延长 LTS 支持窗口，Amazon Linux 也将效仿。
2. 维护支持（第 3-4 年）— 在上游 LTS 支持期结束后，Amazon Linux 团队继续向后移植主要修复关键和重要漏洞 CVEs（CVSS 分数 7.0 及以上）以及已知被利用的漏洞。在此阶段不会向后移植低严重性和中等严重性 CVEs 。

四年后，内核寿命终止，不再接收安全更新。较旧的内核仍可通过存储库使用，您可以继续使用它们。你不应该指望有任何进一步的补丁或修复。我们建议在此日期之前升级到支持的内核。在内核终止支持日期之后，特定于内核的内容 AMIs 将不再更新。

下图显示了当前 Amazon Linux LTS 内核的支持时间表：



内核更新

从 2026 年 6 月开始， AL2023 将每年更新默认内核。这`al2023-ami-kernel-default`组 AMIs 将更新为最新的 LTS 内核，因此新启动的实例将推出新的内核版本。

正在运行的实例不会自动更新到新的内核。要在现有实例上升级内核，必须明确安装新的内核软件包并重新启动。有关更多信息，请参阅 正在更新 Linux 内核 AL2023。

我们强烈建议立即采用新内核，以便从最新的安全性和性能改进中受益。随着时间的推移，较旧的内核收到的安全修复越来越少，速度也更慢。将内核更新整合到您现有的测试和部署管道中，以便在部署到生产环境之前验证兼容性。

发行版之间的内核版本重叠

为了简化 Amazon Linux 发行版之间的迁移，当前和下一版 Amazon Linux 发行版中都将至少有一个内核版本可用。这使您可以先在现有发行版上升级到新内核，验证您的工作负载，然后迁移到新的发行版，因为那里有相同的内核版本。

CVE 处理

AL2023 内核地址 CVEs 如下所示：

- 关键漏洞和重要漏洞 CVEs（CVSS 7.0 及更高版本）以及已知被利用的漏洞将反向移植到所有支持的内核中。
- 在全面支持阶段，通过定期的上游 LTS 变基来解决@@ 低和中 CVEs（CVSS 低于 7.0）的问题。在维护支持阶段， CVEs 它们不会向后移植。如果上游 LTS 变基未在 60 天内解决低或中等 CVE，则在 A mazon Linux 安全中心将该问题标记为 “未计划修复”。

运行最新的可用内核是获取最新安全、性能和功能更新的最佳方式。

你应该做什么

1. 为您的应用程序实施持续集成 (CI) — 在对生产设置进行任何更改之前，请确保在测试阶段对其进行了适当的验证。在验证中包括内核更新。
2. 保持最新内核 — 每款新的年度 LTS 内核一经推出，就立即予以采用。较新的内核可以更快地收到安全修复。使用该`al2023-ami-kernel-default`系列 AMIs 即可自动使用 Amazon Linux 团队推荐的内核版本。
3. 自动更新-使用诸如 S AWS ystems Manager 之类的工具来管理机群中的内核更新。
4. 计划重启-每次内核更新都需要重新启动。在维护计划中加入重启窗口。

AL2023 图形化桌面

自 2023.7 版本起，Amazon Linux 2023 提供了一个基于 GNOME 的可选、轻量级、云优化的图形界面。此现代桌面环境通过内置工具（如用于安全浏览的 Firefox）提供增强的生产力特性，同时保持对 Amazon DCV 和 VNC 远程访问的支持。

> **Note**
>
> AL2023 密切关注上游 Firefox Extended Support Release (ESR) 版本，并会尽快更新到下一个 ESR。有关更多信息，请参阅 Firefox ESR 发布日历和 F irefox 发行说明。

相关主题

有关安装图形桌面环境的更多信息，请参阅以下文档：

- 教程：在上安装 GNOME 桌面环境 AL2023

适用于亚马逊 Linux 的补充软件包

本节介绍了 Amazon Linux (SPAL) 的补充软件包，概述了其优点和局限性，以及报告软件包相关问题的指南。

主题

什么是亚马逊 Linux（或 SPAL）的补充软件包？

适用于 Amazon Linux 的补充包 (SPAL) 是一个专用的软件包存储库，它允许访问源自企业 Linux 9 (EPEL9) 额外软件包的数千个额外软件包。这些软件包补充了核心亚马逊 Linux 2023 中可用的现有软件。

SPAL 通过提供与 AL2 023 兼容的预构建包来简化软件部署，客户无需自己使用源代码构建软件包。这样可以节省软件安装过程中的时间和精力。

> **Note**
>
> SPAL 适用于所有AWS商业区域，包括 AWSGovCloud （美国）地区和中国，适用于发布版本为或更高版本的 AL2 023 实例。 `2023.9.20251117`

优势

SPAL 为亚马逊 Linux 2023 用户提供了几个关键优势：

- 扩展 AL2 023 用例 — 访问核心 AL2 023 存储库之外的热门其他软件包，例如 `GDAL`或 `pandocdrbd-utils`，使客户能够支持多种业务和开发需求。

- 简化 AL2 023 包管理 — 通过提供为 AL2 023 预先构建的软件包，消除了从源代码构建其他包的过程，从而节省了时间并降低了编译错误的风险。
- 简化 AL2 到 AL2 023 迁移 — SPAL 存储库支持将工作负载从 023 无缝迁移 AL2 到 AL2 023，包括依赖包的工作负载。 EPEL7

Support 对 SPAL 软件包的支持

SPAL 软件包的支持级别与核心 AL2 023 软件包不同，后者在 Amazon Linux 2023 的整个生命周期内都获得支持。

> **Important**
>
> 在使用 SPAL 之前，客户必须仔细评估以下注意事项：
>
> - Su AWS pport P lans 不包括 SPAL 套餐。
> - SPAL 软件包是从上游 “按原样” 提供的。 EPEL9
> - SPAL 包裹将不会收到 AWS CVE 安全追踪信息。
> - SPAL 软件包仅从上游获得安全补丁和错误修复（ EPEL9 如果有）。

报告与包裹相关的问题

如果您遇到 SPAL 软件包的问题，我们建议您先检查上游 EPEL9 存储库中的相应软件包是否出现相同的问题。为此，请查阅上游 EPEL 存储库中的问题列表。

如果中不存在问题 EPEL9，请在 Amazon Linux 2023 GitHub 存储库中创建问题，因为这表明问题特定于 SPAL 软件包版本或配置。

这种方法可确保问题由适当的维护者解决，并有助于提高 SPAL 和上游 EPEL 软件包的整体质量。

> **Note**
>
> 报告的问题将尽力处理。

相关主题

有关在系统上配置 SPAL 的信息，请参阅以下文档页面：

教程：在 023 上 AL2配置 SPAL 存储库

适用于 Amazon Linux 的补充软件包 (SPAL) 是 AL2 023 的附加软件包存储库，可让客户访问成千上万的开源软件包。

以下教程可帮助您在 AL2 023 实例上配置 SPAL 存储库。通过安装存储库，您将可以访问 SPAL 中所有可用的 RPM 软件包。安装完成后，您可以使用软件包管理器在系统上安装和使用这些软件包。

内容

前提条件

本教程假设您已经使用 AL2 023 发行版`2023.9.20251117`或更高版本启动了实例。更多信息，请参阅 Amazon EC2 上的 AL2023 和 正在更新 AL2023 页面。

检查先决条件

- 要验证您的实例是否满足先决条件，您可以检查系统上`system-release`安装的版本。

  要检查软件包的版本，可以使用以下命令。

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

  该命令将显示有关软件包的信息，包括主要版本。

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

> **Note**
>
> 确保`system-release`安装了最新版本的。你可以运行更新`sudo dnf upgrade`到最新版本。

在您的系统上安装 SPAL

1. 在您的系统上安装该`spal-release`软件包。这会将`.repo`配置文件和 GPG 密钥添加到您的系统中。

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

> **Note**
>
> 在安装过程中，将显示支持声明。该声明解释了SPAL的支持范围和限制。请花点时间仔细查看此信息。

2. 验证 SPAL 存储库配置已成功添加到您的系统中。

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

您应该看到系统上配置的两个存储库：`amazonlinux-spal`和 `amazonlinux-spal-source`

您也可以通过运行来查看已配置的存储库列表`dnf repolist`。

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

> **Note**
>
> 必须使用该`--all`标志才能同时查看已启用和禁用的存储库。

两个 SPAL 存储库都应可用。请注意，亚马逊 Linux 2023 SPAL 存储库-源包存储库在默认情况下处于禁用状态。

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

3. （可选）启用源存储库。

> **Note**
>
> 默认情况下，RPM 源 (SRPM) 存储库通常处于禁用状态，因为开发人员主要使用它们来构建软件包，而不是由最终用户用于安装软件。当您使用需要源包的命令时，DNF 会自动启用源存储库，例如`dnf download --source package`。
> 您无需手动启用源存储库即可进行一次性源包操作。仅当要在系统上通过 SPAL 进行重建 SRPMs 时，才执行此步骤。

要永久启用系统上的 Amazon Linux 2023 SPAL 存储库-源包存储库，请运行以下命令：

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

正在安装 SPAL 软件包

- 通过运行`dnf install`命令在系统上安装 SPAL 软件包。

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

> **Note**
>
> 您可以使用`dnf list`查看 SPAL 软件包的完整列表。
>
> ```
> [ec2-user ~]$ dnf list --repo=amazonlinux-spal
> ```

> **Note**
>
> SPAL 是一个版本控制存储库。确保`system-release`安装了最新版本的，以查看最新的软件包列表。
> 有关确定性更新的更多信息，可以查看 通过版本控制的存储库进行确定性升级 AL2023

从系统中卸载 SPAL 存储库

1. 使用`dnf remove`命令删除 SPAL 存储库配置。

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

2. 通过运行`dnf repolist`命令验证存储库是否已删除。

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

> **Important**
>
> 从系统中删除 SPAL 存储库配置不会删除系统上安装的任何 SPAL 软件包。

相关主题

有关 Amazon Linux 补充包存储库的更多信息，请参阅以下文档：

- 适用于亚马逊 Linux 的补充软件包

在 AL2023 上运行应用程序

这部分介绍在 Amazon Linux 2023 (AL2023) 上运行应用程序的方法，包括管理它们的启动（和重启）时间以及控制资源使用量。

主题

在 AL2023 中使用 systemd 限制进程资源使用量

在 Amazon Linux 2023（AL2023）上，我们推荐使用 systemd 来控制进程或进程组可以使用的资源。使用 systemd 是一个强大且易于使用的替代方案，可以替代手动操作 cgroups 或使用诸如 cpulimit 之类的工具，后者先前仅在第三方 EPEL 存储库中可用于 Amazon Linux。

有关全面信息，请参阅上游 systemd 关于 systemd.resource-control 的文档，或在 AL2023 实例上查看 systemd.resource-control 的 man 手册页。

以下示例将使用 stress-ng CPU 压力测试（来自 stress-ng 程序包）来模拟 CPU 密集型应用程序，并使用 memcached 来模拟内存密集型应用程序。

以下示例涵盖对一次性命令施加 CPU 限制和对服务施加内存限制。systemd 提供的大多数资源约束可以在任何 systemd 运行进程的地方使用，并且可以同时使用多个约束。为了说明目的，以下示例仅限于单一约束。

使用 **systemd-run** 对运行一次性命令进行资源控制

虽然通常与系统服务关联，但非 root 用户也可使用 systemd 来运行服务、调度定时器或运行一次性进程。在以下示例中，我们将使用 stress-ng 作为示例应用程序。在第一个示例中，我们将使用 systemd-run 在 ec2-user 默认账户中运行它；在第二个示例中，我们将对其 CPU 使用量施加限制。

Example在命令行使用 systemd-run 运行进程，不限制资源使用量

1. 确保已安装 stress-ng 程序包，因为我们将以其为例。

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

2. 使用 systemd-run 执行一个 10 秒的 CPU 压力测试，不限制其可使用的 CPU 量。

```
[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
```

--user 选项指示 systemd-run 以当前登录用户身份执行命令，--tty 选项表示附加一个 TTY，--wait 表示等待服务完成，--property=CPUAccounting=1 选项指示 systemd-run 记录运行该进程所使用的 CPU 时间。--property 命令行选项可用于传递可在 systemd.unit 配置文件中配置的 systemd-run 设置。

当被指示对 CPU 施加负载时，stress-ng 程序将在您要求的运行时间内使用所有可用的 CPU 时间执行其测试。对于实际应用，可能需要对进程的总运行时间设置限制。在以下示例中，我们将要求 stress-ng 运行的时间长于我们使用 systemd-run 为其设置的最长持续时间限制。

Example在命令行使用 systemd-run 运行进程，将 CPU 使用量限制为 1 秒

1. 确保已安装 stress-ng 以运行此示例。
2. LimitCPU 属性等效于 ulimit -t，它将限制该进程允许使用的 CPU 最长时间。在这种情况下，由于我们要求进行 10 秒的压力运行，但将 CPU 使用量限制为 1 秒，该命令将收到 SIGXCPU 信号并失败。

```
[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
```

更常见的情况是，您可能希望限制特定进程可消耗的 CPU 时间百分比。在以下示例中，我们将限制 `stress-ng` 可消耗的 CPU 时间百分比。对于实际的服务，可能需要限制后台进程可消耗的 CPU 时间最大百分比，以便为处理用户请求的进程空出资源。

Example使用 systemd-run 将进程限制在单颗 CPU 10% 的 CPU 时间

1. 确保已安装 `stress-ng` 以运行此示例。
2. 我们将使用 `CPUQuota` 属性来告知 `systemd-run` 约束即将运行的命令的 CPU 使用量。我们不限制进程的运行时间，只限制其可以使用的 CPU 量。

```
[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
```

请注意 CPU 统计信息显示，虽然服务运行了 10 秒，但它仅消耗了 1 秒的实际 CPU 时间。

有多种方法可以配置 `systemd` 来限制 CPU、内存、网络和 IO 的资源使用量。有关完整文档，请参阅上游 `systemd` 关于 systemd.resource-control 的文档，或在 AL2023 实例上查看 `systemd.resource-control` 的 man 手册页。

在底层，`systemd` 利用 Linux 内核的特性（如 `cgroups`）来实现这些限制，同时避免了您手动配置的需要。Linux 内核关于 `cgroup-v2` 的文档包含了关于 `cgroups` 工作的详尽细节。

在 systemd 服务中进行资源控制

有多个参数可以添加到 systemd 服务的 [Service] 部分，以控制系统资源使用量。这些参数包括硬限制和软限制。关于每个选项的确切行为，请参考上游 systemd 关于 systemd.resource-control 的文档，或在 AL2023 实例上查看 systemd.resource-control 的 man 手册页。

常用的限制包括：使用 MemoryHigh 指定内存使用的节流限制，使用 MemoryMax 设置硬性上限（一旦达到，将调用 OOM Killer），以及使用 CPUQuota（如前一节所示）。也可以配置权重和优先级，而不是固定的数值。

Example使用 systemd 为服务设置内存使用限制

在本示例中，我们将为 memcached（一个简单的键值缓存）设置内存使用硬限制，并展示 OOM Killer 是如何针对该服务而非整个系统被调用的。

1. 首先，我们需要安装此示例所需的程序包。

```
[ec2-user ~]$ sudo dnf install -y memcached libmemcached-awesome-tools
```

2. 启用 memcached.service 然后启动服务，使 memcached 运行。

```
[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. 检查 memcached.service 是否正在运行。

```
[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. 既然 memcached 已安装并运行，我们可以通过向缓存中插入一些随机数据来观察其功能

   在 /etc/sysconfig/memcached 中，CACHESIZE 变量默认设置为 64，即 64 兆字节。通过向缓存中插入超过最大缓存大小的数据，我们可以看到缓存被填满，并且使用 memcached-tool 驱逐了一些项目，同时 memcached.service 使用了大约 64MB 的内存。

```
[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. 使用 MemoryMax 属性为 memcached.service 设置一个硬限制，如果达到此限制，将调用 OOM Killer。可以通过将其他选项添加到覆盖文件来为服务设置它们。这可以通过直接编辑 /etc/systemd/system/memcached.service.d/override.conf 文件或使用 systemctl 的 edit 命令交互式完成。

```
[ec2-user ~]$ sudo systemctl edit memcached.service
```

   将以下内容添加到覆盖文件中，为服务设置 32MB 内存的硬限制。

```
[Service]
MemoryMax=32M
```

6. 告诉 systemd 重新加载其配置

```
[ec2-user ~]$ sudo systemctl daemon-reload
```

7. 观察 memcached.service 现在正以 32MB 的内存限制运行。

```
[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. 在使用少于 32MB 内存时，服务将正常运行，我们可以通过向缓存加载少于 32MB 的随机数据，然后检查服务状态来验证。

```
[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. 我们现在可以通过尝试使用默认 memcached 配置所允许的完整 64MB 缓存，使 memcached 使用超过 32MB 的内存。

```
[ec2-user ~]$ for i in $(seq 1 150); do dd if=/dev/random of=$i bs=512k count=1;
 memcp -s localhost $i; done
```

您将观察到在上述命令的某个时刻，会出现连接到 memcached 服务器的错误。这是因为 OOM Killer 由于我们施加的限制而终止了该进程。系统的其余部分将正常运行，OOM Killer 不会考虑其他进程，因为我们只限制了 memcached.service。

```
[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
 ${CACHESIZE} -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'.
```

在 AL2023 中使用 cgroups 限制进程资源使用量

虽然推荐使用 使用 systemd 进行资源控制，但这部分介绍使用基础 libcgroup-tools 工具限制进程 CPU 和内存使用量的基本用法。这两种方法都是使用 cpulimit 工具（先前位于 EPEL 中）的替代方案。

以下示例涵盖在运行 stress-ng 压力测试（来自 stress-ng 程序包）的同时，使用 libcgroup-tools 程序包中的工具以及 sysfs 中的可调参数来限制其 CPU 和内存使用量。

在命令行使用 **libcgroup-tools** 限制资源使用量

1. 安装 libcgroup-tools 软件包。

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

2. 创建一个包含 memory 和 cpu 控制器的 cgroup，并为其命名 (our-example-limits)。使用 -a 和 -t 选项允许 ec2-user 用户控制 cgroup 的可调参数

```
[ec2-user ~]$ sudo cgcreate -a ec2-user -t ec2-user -g memory,cpu:our-example-limits
```

现在存在一个 /sys/fs/cgroup/our-example-limits/ 目录，其中包含可用于控制每个可调参数的文件。

> **Note**
>
> Amazon Linux 2 使用 cgroup-v1，而非 AL2023 使用的 cgroup-v2。在 AL2 上，sysfs 路径不同，并且会存在由 ec2-user 拥有的 /sys/fs/cgroup/memory/our-example-limits 和 /sys/fs/cgroup/cpu/our-example-limits 目录，这些目录包含可用于控制 cgroup 限制的文件。

3. 将我们 cgroup 中所有进程的内存使用量限制为 1 亿字节。

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

> **Note**
>
> Amazon Linux 2 使用 cgroup-v1，而非 Amazon Linux 2023 使用的 cgroup-v2。这意味着某些可调参数是不同的。要在 AL2 上限制内存使用量，需改用以下可调参数。

```
[ec2-user ~]$ echo 10000000 > /sys/fs/cgroup/memory/our-example-limits/
memory.limit_in_bytes
```

4. 将我们 cgroup 中所有进程的 CPU 使用量限制为 10%。cpu.max 文件的格式为 $MAX $PERIOD，将组限制为每 $PERIOD 周期内消耗 $MAX。

```
[ec2-user ~]$ echo 10000 100000 > /sys/fs/cgroup/our-example-limits/cpu.max
```

Amazon Linux 2 使用 cgroup-v1，而非 Amazon Linux 2023 使用的 cgroup-v2。这意味着某些可调参数是不同的，包括限制 CPU 使用量的方式。

5. 以下示例在 our-example-limits cgroup 中运行 stress-ng（可通过运行 dnf install -y stress-ng 安装）。当 stress-ng 命令运行时，您可以使用 top 观察到其被限制在 CPU 时间的 10%。

```
[ec2-user ~]$ sudo cgexec -g memory,cpu:our-example-limits stress-ng --cpu 1
```

6. 通过移除 cgroup 进行清理

```
[ec2-user ~]$ sudo cgdelete -g memory,cpu:our-example-limits
```

Linux 内核关于 cgroup-v2 的文档包含关于其工作原理的详细说明。cpu 和 memory 控制器的文档涵盖了如何使用每个可调选项的细节。

AL2023 在上使用 AWS

您可以设置 AL2023 为与其他人一起使用 AWS 服务。例如，您可以在启动亚马逊弹性计算云 (Amazon EC2) 实例时选择 AM AL2023 I。

对于这些设置过程，您可以使用 AWS Identity and Access Management (IAM) 服务。有关 IAM 的完整信息，请参阅以下参考材料：

- AWS Identity and Access Management (IAM)
- IAM 用户指南

主题

入门 AWS

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS 管理控制台在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 Signing in as the root user。
2. 为您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录 URL。

有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Create a permission set。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Add groups。

授权以编程方式访问

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS 管理控制台。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	（推荐）使用控制台凭证作为临时凭证，签署对 AWS CLI AWS SDKs、或的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》中的“登录 AWS 本地开发”。 • 有关信息 AWS SDKs，请参阅《工具参考指南》AWS SDKs 和《工具参考指南》中的登录进行 AWS 本地开发。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关工具和 AWS SDKs AWS APIs，请参阅《工具参考指南》中的 IAM 身份中心身份验证AWSSDKs 和工具参考指南。
IAM	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDKs 和工具，请参阅《工具参考指南》AWS SDKs 和《工具参考指南》中的使用长期凭证进行身份验证。 • 有关信息 AWS APIs，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

Amazon EC2 上的 AL2023

使用以下过程之一启动带有 AL2023 AMI 的 Amazon EC2 实例。您可以选择标准 AMI，也可以选择 AMI 最低版本。有关标准 AMI 与AMI 最低版本 之间区别的更多信息，请参阅比较 AL2023 标准（默认）和最小 AMIs。

主题

使用 Amazon EC2 控制台启动 AL2023

使用亚马逊 EC2 控制台启动 AL2023 AMI。

> **Note**
>
> 对于基于 ARM 的实例， AL2023 仅支持使用 Graviton2 或更高版本处理器的实例类型。AL2023 不支持 A1 实例。

使用以下步骤，从 Amazon EC2 控制台启动具有 AL2023 AMI 的 Amazon EC2 实例。

使用 AL2023 AMI 启动 EC2 实例

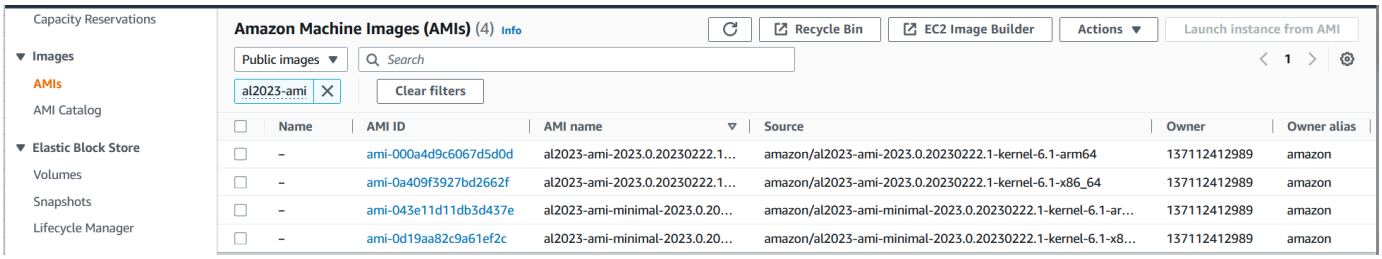
1. 打开位于 https://console.aws.amazon.com/ec2/ 的 Amazon EC2 控制台。
2. 在导航窗格中，请选择 AMIs。
3. 从下拉菜单中选择公有映像。
4. 在搜索字段中输入 **al2023-ami**。

> **Note**
>
> 确保 amazon 显示在拥有者别名列中。

5. 从列表中选择一个映像。在来源下，您可以确定使用标准 AMI 还是AMI 最低版本。 AL2023 AMI 名称可以使用以下格式进行解释：

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

6. 下图显示了部分列表 AL2023 AMIs。



有关如何启动 Amazon EC2 实例的信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 Linux 实例入门。

AL2023 使用 SSM 参数启动和 AWS CLI

在中 AWS CLI，您可以使用 AMI 的 SSM 参数值启动的新实例。 AL2023更具体地说，使用以下列表中的一个动态 SSM 参数值，然后在 SSM 参数 value/ 之前添加 `/aws/service/ami-amazon-linux-latest/`。您可以使用以下参数在 AWS CLI中启动实例。

- `al2023-ami-kernel-default-arm64`，适用于 arm64 架构
- `al2023-ami-minimal-kernel-default-arm64`，适用于 arm64 架构（AMI 最低版本）
- `al2023-ami-kernel-default-x86_64`，适用于 x86_64 架构
- `al2023-ami-minimal-kernel-default-x86_64`，适用于 x86_64 架构（AMI 最低版本）

> **Note**
>
> 每个*`italic`*项目都是一个示例参数。将它们替换为您自己的信息。

```
$ 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
```

`--image-id` 标志指定 SSM 参数值。

`--instance-type` 标志指定实例的类型和大小。该标志必须与您选择的 AMI 类型兼容。

该`--region`标志指定您在 AWS 区域 哪里创建实例。

该`--key-name`标志指定 AWS 区域了用于连接到实例的密钥。如果您不提供创建实例的区域中存在的密钥，则无法使用 SSH 连接到该实例。

`--security-group-ids` 标志指定可确定对入站和出站网络流量的访问权限的安全组。

> **Important**
>
> AWS CLI 要求您指定允许通过端口从远程计算机访问实例的现有安全组TCP:22。如果没有指定的安全组，您的新实例将被置于默认安全组中。在默认安全组中，您的实例只能与您的 VPC 中的其他实例连接。

有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的启动、列出和终止 EC2 实例。

使用启动最新的 AL2023 AMI CloudFormation

要使用启动 AL2023 AMI CloudFormation，请使用以下模板之一。

> **Note**
>
> `x86_64`和`Arm64` AMIs 都需要不同的实例类型。有关更多信息，请参阅 Amazon EC2 实例类型。

`JSON` 模板：

```
{
  "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 模板：

```
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
```

如果需要，请务必替换“Default”部分末尾的 AMI 参数。可以使用以下参数值：

- `al2023-ami-kernel-6.1-arm64`，适用于 arm64 架构
- `al2023-ami-minimal-kernel-6.1-arm64`，适用于 arm64 架构（AMI 最低版本）
- `al2023-ami-kernel-6.1-x86_64`，适用于 x86_64 架构

- `al2023-ami-minimal-kernel-6.1-x86_64`，适用于 x86_64 架构（AMI 最低版本）

以下是动态内核规范。每次主要内核版本更新时，默认内核版本都会自动更改。

- `al2023-ami-kernel-default-arm64`，适用于 arm64 架构
- `al2023-ami-minimal-kernel-default-arm64`，适用于 arm64 架构（AMI 最低版本）
- `al2023-ami-kernel-default-x86_64`，适用于 x86_64 架构
- `al2023-ami-minimal-kernel-default-x86_64`，适用于 x86_64 架构（AMI 最低版本）

AL2023 使用特定的 AMI ID 启动

您可以使用 AMI ID 启动特定的 AL2023 AMI。您可以通过查看 Amazon EC2 控制台中的 AMI 列表来确定需要哪个 AL2023 AMI ID。或者，你可以使用 AWS Systems Manager。如果您使用的是 Systems Manager，请务必从上一节中列出的别名中选择 AMI 别名。有关更多信息，请参阅 IDs 使用 AWS Systems Manager 参数存储查询最新的 Amazon Linux AMI。

AL2023 AMI 的弃用和生命周期

每个新 AL2023 版本都包含一个新的 AMI。AMI 注册后，会标记有弃用日期。每个 AL2023 AMI 的弃用日期为自发布之日起 90 天，AL2023 上的内核实时修补以匹配为每个内核版本提供的时间段。

> **Note**
>
> 90 天弃用日期是指单个 AMI，并不指 AL2023 发布频率或产品支持期。

有关 AMI 弃用的更多信息，请参阅《Amazon EC2 用户指南》中的弃用 AMI。

定期使用更新的 AMI 启动实例，可确保实例启动时有最新的安全更新，包括更新的内核。如果您启动先前版本的 AMI 并应用更新，则该实例在一段时间内会没有最新的安全更新。为确保您使用的是最新的 AMI，建议您使用 SSM 参数。

有关如何使用 SSM 参数启动实例的更多信息，请参阅：

- AL2023 使用 SSM 参数启动和 AWS CLI
- 使用启动最新的 AL2023 AMI CloudFormation

连接到 AL2023 实例

使用 SSH 或 AWS Systems Manager 连接到您的 AL2023 实例。

使用 SSH 连接到您的实例

有关如何使用 SSH 连接到 Linux 实例的说明，请参阅《Amazon EC2 用户指南》中的使用 SSH 连接到 Linux 实例。

使用 Connect 连接到您的实例 AWS Systems Manager

有关 AWS Systems Manager 如何使用连接 AL2023 实例的说明，请参阅 Amazon EC2 用户指南中的使用会话管理器连接您的 Linux 实例。

使用 Amazon EC2 Instance Connect

AL2023 AMI（不包括 AMI 最低版本）默认安装了 EC2 Instance Connect 代理。要将 EC2 Instance Connect 与从最小 AMI 启动的 AL2023 实例一起使用，您必须安装该`ec2-instance-connect`软件包。有关使用 EC2 Instance Connect 的说明，请参阅《Amazon EC2 用户指南》中的使用 EC2 Instance Connect 连接到 Linux 实例。

比较 AL2023 标准和最小值 AMIs

您可以使用标准（默认）或最低 AL2023 AMI 版本启动 Amazon EC2 实例。有关如何启动标准或 AMI 最低版本类型的 Amazon EC2 实例的说明，请参阅 Amazon EC2 上的 AL2023。

标准 AL2023 AMI 安装了所有最常用的应用程序和工具。如果您想快速入门并且对自定义 AMI 不感兴趣，建议您使用标准 AMI。

最小 AL2023 AMI 是简化的基本版本，仅包含运行操作系统 (OS) 所需的最基本的工具和实用程序。如果您希望操作系统占用空间尽可能小，建议您使用AMI 最低版本。AMI 最低版本 可以稍微降低磁盘空间占用，提高长期成本效益。如果您想要更小的操作系统并且不介意手动安装工具和应用程序，则AMI 最低版本 是合适的选项。

容器镜像更接近包集中的 AL2023 最小 AMI。

比较 Amazon Linux 2023 映像上安装的软件包

AMI、Minimal AL2023 AMI 和容器镜像上的 RPMs 当前镜像的比较。

程序包	AMI	AMI 最低版本	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		
audit	3.0.6	3.0.6	
audit-libs	3.0.6	3.0.6	3.0.6

程序包	AMI	AMI 最低版本	Container
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	2.1.1		
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		
checkpolicy	3.4	3.4	
chkconfig	1.15		

程序包	AMI	AMI 最低版本	Container
chrony	4.3	4.3	
cloud-init	22.2	22.2	
cloud-init-cfg-ec2	22.2	22.2	
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	
curl-minimal	8.5.0	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27	

程序包	AMI	AMI 最低版本	Container
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		
dracut	055	055	

程序包	AMI	AMI 最低版本	Container
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	

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	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-livepatch-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

程序包	AMI	AMI 最低版本	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		

程序包	AMI	AMI 最低版本	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-hmaccalc	1.4.0	1.4.0	

程序包	AMI	AMI 最低版本	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

程序包	AMI	AMI 最低版本	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_nss_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		

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	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-libs	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	

程序包	AMI	AMI 最低版本	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		

程序包	AMI	AMI 最低版本	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		

程序包	AMI	AMI 最低版本	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		

程序包	AMI	AMI 最低版本	Container
policycoreutils	3.4	3.4	
policycoreutils-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-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	

程序包	AMI	AMI 最低版本	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	

程序包	AMI	AMI 最低版本	Container
python3-jsonschema	3.2.0	3.2.0	
python3-libcomps	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-libselinux	3.4	3.4	
python3-libsemanage	3.4	3.4	
python3-libstoragemgmt	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	21.3.1
python3-ply	3.11	3.11	
python3-policycoreutils	3.4	3.4	

程序包	AMI	AMI 最低版本	Container
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	
python3-requests	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-setuptools	59.6.0	59.6.0	

程序包	AMI	AMI 最低版本	Container
python3-setuptools-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	

程序包	AMI	AMI 最低版本	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		

程序包	AMI	AMI 最低版本	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

程序包	AMI	AMI 最低版本	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		
xfsprogs	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

程序包	AMI	AMI 最低版本	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	

AL2023 在容器中使用

> **Note**
>
> 有关如何使用在 Amazon ECS AL2023 上托管容器化工作负载的更多信息，请参阅。AL2023 for Amazon ECS 容器托管

根据用例，有几种方法 AL2023 可以在容器内部使用。与AL2023 基本容器镜像亚马逊 Linux 2 容器镜像和最小的 AMI AL2023 最为相似。

对于高级用户，我们提供了在 AL2023 .2 版本中引入的最小容器镜像，以及描述如何构建基本容器的文档。

AL2023 也可以用来托管容器化工作负载，既可以是 AL2023基于容器镜像，也可以是基于其他 Linux 发行版的容器。您可以使用 AL2023 for Amazon ECS 容器托管，也可以直接使用提供的容器运行时包。`docker`、`containerd`、和`nerdctl`软件包可供安装和使用 AL2023。

主题

- 使用 AL2023 基本容器镜像
- AL2023 最小容器镜像
- 构建基本容器镜像 AL2023
- 比较 Amazon Linux 2023 容器映像上安装的软件包
- 比较 Amazon Linux 2023 AMI 最低版本 和容器映像上安装的软件包

使用 AL2023 基本容器镜像

AL2023 容器映像由 AL2023 AMI 中包含的相同软件组件构建。作为 Docker 工作负载的基本映像，它可用在任何环境中。如果您在 Amazon Elastic Compute Cloud (Amazon EC2) 中针对应用程序使用 Amazon Linux AMI，就可以使用 Amazon Linux 容器映像将您的应用程序容器化。

在本地开发环境中使用 Amazon Linux 容器镜像，然后使用亚马逊弹性容器服务 (Amazon ECS) 将您的应用程序推送到 AWS 使用亚马逊弹性容器服务 (Amazon ECS)。有关更多信息，请参阅《Amazon Elastic Container Registry 用户指南》中的将 Amazon ECR 映像与 Amazon ECS 结合使用。

Amazon Linux 容器映像在 Amazon ECR Public 上可用。您可以 AL2023 通过您的指定 AWS 代表提供反馈，也可以通过在 ama zon-linux- 2023 存储库中提交问题来提供反馈。 GitHub

从 Amazon ECR Public 中提取 Amazon Linux 容器映像

1. 对您的 Amazon Linux Public 注册表进行 Docker 客户端身份验证。验证令牌的有效期为 12 小时。有关更多信息，请参阅《Amazon Elastic Container Registry 用户指南》中的私有注册表身份验证。

   > **Note**
   >
   > 最新版本的 AWS CLI 版本 2 支持该get-login-password命令。有关更多信息，请参阅《AWS Command Line Interface 用户指南》中的安装 AWS Command Line Interface。

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

   输出如下所示。

   ```
   Login succeeded
   ```

2. 运行 docker pull 命令以拉取 Amazon Linux 容器映像。要在 Amazon ECR 公开映像浏览馆中查看 Amazon Linux 容器映像，请参阅 Amazon ECR 公开映像浏览馆 - amazonlinux。

   > **Note**
   >
   > 拉取 AL2023 Docker容器镜像时，您可以使用以下格式之一的标签：

- 要获取最新版本的 AL2023 容器镜像，请使用:2023标签。
- 要获取的特定版本 AL2023，可以使用以下格式：
  - :2023.*[0-7 release quarter].[release date].[build number]*

以下示例使用标签:2023并提取最新的可用容器映像 AL2023。

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

3. （可选）在本地运行容器。

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

从 Docker Hub 提取 AL2023 容器镜像

1. 使用docker pull命令拉取 AL2023 容器镜像。

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

2. （可选）在本地运行容器。

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

> **Note**
>
> 的容器映像仅 AL2023 使用软件dnf包管理器来安装软件包。这意味着没有 amazon-linux-extras 或等效的命令可以用于其他软件。

AL2023 最小容器镜像

> **Note**
>
> 标准 AL2023 容器镜像适用于大多数用例，适应最小的容器镜像可能比适应 AL2023 基本容器镜像要花更多的精力。

AL2023.2 中引入 AL2023 的最小容器镜像与基础容器镜像不同，因为它只包含安装其他软件包所需的最低限度的软件包。最小容器映像被设计为最小程序包集合，而非便捷程序包集合。

AL2023 最小容器镜像由中已有的软件组件构建 AL2023。最小容器映像的关键区别在于使用 `microdnf` 来提供 `dnf` 程序包管理器，而非功能齐全的基于 Python 的 `dnf`。这样可以缩小最小容器映像，但要权衡一下没有dnf包管理器的完整功能集（包含在基础容器镜像中 AL2023 AMIs ）。

最 AL2023 小的容器镜像构成了 `provided.al2023` AWS Lambda 运行时环境的基础。

有关最小容器映像中包含的程序包的详细列表，请参阅 比较 Amazon Linux 2023 容器映像上安装的软件包。

最小容器映像大小

由于 AL2023 最小容器镜像包含的包比 AL2023基础容器镜像少，因此它也要小得多。下表比较了当前和过去版本的 Amazon Linux 的容器映像选项。

> **Note**
>
> 映像大小如 Amazon ECR 公开映像浏览馆上的 Amazon Linux 所示。

Image	版本	映像大小	备注
亚马逊 Linux (1AL1)	2018.03.0.20230918.0	62.3MB	仅限 `x86-64`
Amazon Linux 2	2.0.20230926.0	64.2MB	`aarch64` 比 `x86-64` 大 1.6MB

Image	版本	映像大小	备注
Amazon Linux 2023 基本容器映像	2023.2.20231002.0	52.4MB	
Amazon Linux 2023 最小容器映像	2023.2.20231002.0-minimal	35.2MB	

使用 AL2023 最小容器镜像

AL2023 最小容器镜像在上ECR可用，`2023-minimal`标签将始终指向 AL2023 基于最新版本的最小容器镜像，而`minimal`标签可能会更新为更新的 Amazon Linux 版本 AL2023。

您可以使用 `docker` 拉取这些标签，示例如下：

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

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

以下示例展示了采用最小容器映像并在其上安装 GCC 的 `Dockerfile` 的示例：

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2023-minimal
RUN dnf install -y gcc && dnf clean all
```

构建基本容器镜像 AL2023

构建 AL2023 容器镜像的软件组件与 AL2023 AMI 中包含的软件组件相同。它包含一种软件，使基本容器层能够表现出类似于在 Amazon EC2 实例上运行的行为，例如程序包管理器 `dnf`。这部分介绍如何从头构建一个仅包含应用程序所需最低限度依赖项的容器。

> **Note**
>
> 标准 AL2023 容器镜像适用于大多数用例。使用标准容器映像可以轻松地在映像之上进行构建。精简版容器映像会使得在您的映像基础上进行构建变得更加困难。

针对应用程序创建具有最少依赖项的容器

1. 确定运行时依赖项。这将因您的应用程序而异。
2. 构造一个构建 FROM scratch 的 Dockerfile/Containerfile。Dockerfile 的以下示例可用于构建仅包含 bash shell 及其依赖项的容器。

```
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"]
```

- Dockerfile 的运行方式是：

  1. 启动名为的 AL2023 容器build。该容器将用于引导基本容器，该容器本身不会部署，而会生成要部署的容器。
  2. 创建 /sysroot 目录。该目录将是 build 容器安装基本容器所需的依赖项的地方。在接下来的步骤中，/sysroot 路径将被打包为我们的基本映像的根目录。

     以这种dnf方式使用--installroot选项就是我们创建其他 AL2023 图像的方式。dnf 的一项功能使得安装程序和映像创建工具能够正常工作。
  3. 调用 dnf 以将软件包安装到 /sysroot。

     rpm -q system-release --qf '%{VERSION}' 命令会查询 (-q) system-release 软件包，设置查询格式 (--qf) 以打印出所查询的软件包的版本（ %{VERSION} 变量是 RPM 版本的 rpm 变量 ）。

     通过将 dnf 的 --releasever 参数设置为 build 容器中的 system-release 版本，每当发布更新的 Amazon Linux 容器基本映像时，都可以使用 Dockerfile 来重建基本容器。

可以将其设置为任何亚马逊 Linux 2023 版本，例如 2023.10.20260216。--releasever这样做意味着build容器将以最新 AL2023版本运行，但无论当前版本如何，都要从 2023.10.20260216 开始构建准系统容器。 AL2023

--setopt=install_weak_deps=False 配置选项可告诉 dnf 只安装必需 的依赖项，而不是推荐或建议的依赖项。

4. 将已安装的系统复制到空白 (FROM scratch) 容器的根目录中。
5. 在本例 /bin/bash 中，将 ENTRYPOINT 设置为所需的二进制文件。

3. 创建一个空目录，将步骤 2 中示例的内容添加到名为 Dockerfile 的文件中。

```
$ 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. 通过运行以下命令构建容器。

```
$ docker build -t al2023-barebones-bash-example
```

5. 使用以下命令运行容器，了解仅限 bash 的最小容器的运行方式。

```
$ 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
```

举一个更实际的例子，以下过程可为显示 Hello World! 的一个 C 应用程序构建一个容器。

1. 创建一个空目录并添加 C 源代码和 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. 使用以下命令构建容器。

```
$ docker build -t al2023-barebones-c-hello-world-example .
```

3. 使用以下命令运行容器。

```
$ docker run -it --rm al2023-barebones-c-hello-world-example
Hello World!
```

比较 Amazon Linux 2023 容器映像上安装的软件包

AL2023 基础容器镜像上的 RPMs 当前图像与 AL2023 最小容器镜像上的 RPMs 当前图像的比较。

程序包	Container	最小容器
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

程序包	Container	最小容器
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
gdbm-libs	1.19	
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-common	2.34	2.34
glibc-minimal-langpack	2.34	2.34
gmp	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7
gobject-introspection		1.73.0
gpgme	1.15.1	1.15.1
grep	3.8	3.8
json-c	0.14	0.14

程序包	Container	最小容器
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
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

程序包	Container	最小容器
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-filesystem	2.15.2	2.15.2
libselinux	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
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

程序包	Container	最小容器
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
openssl-libs	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	

程序包	Container	最小容器
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
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

比较 Amazon Linux 2023 AMI 最低版本 和容器映像上安装的软件包

将 Minimi AL2023 al AMI 上的 RPMs 当前图像与 AL2023 基础镜像和最小容器镜像上的 RPMs 当前图像进行比较。

程序包	AMI 最低版本	Container	最小容器
alternatives	1.15	1.15	1.15
amazon-chrony-config	4.3		
amazon-ec2-net-utils	2.5.1		
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

程序包	AMI 最低版本	Container	最小容器
checkpolicy	3.4		
chrony	4.3		
cloud-init	22.2		
cloud-init-cfg-ec2	22.2		
cloud-utils-growpart	0.31		
coreutils	8.32		
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		

程序包	AMI 最低版本	Container	最小容器
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
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		

程序包	AMI 最低版本	Container	最小容器
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	
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

程序包	AMI 最低版本	Container	最小容器
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
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		

程序包	AMI 最低版本	Container	最小容器
grub2-tools-minimal	2.06		
grubby	8.40		
gzip	1.12		
hostname	3.23		
hwdata	0.384		
inih	49		
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		

程序包	AMI 最低版本	Container	最小容器
kernel-livepatch-repo-s3	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
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		

程序包	AMI 最低版本	Container	最小容器
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
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-hmaccalc	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

程序包	AMI 最低版本	Container	最小容器
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		
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

程序包	AMI 最低版本	Container	最小容器
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		
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

程序包	AMI 最低版本	Container	最小容器
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		
openssl-libs	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-libs	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

程序包	AMI 最低版本	Container	最小容器
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		
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	

程序包	AMI 最低版本	Container	最小容器
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		
python3-jmespath	0.10.0		
python3-jsonpatch	1.21		
python3-jsonpointer	2.0		
python3-jsonschema	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-libselinux	3.4		
python3-libsemanage	3.4		

程序包	AMI 最低版本	Container	最小容器
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		
python3-policycoreutils	3.4		
python3-prettytable	0.7.2		
python3-prompt-toolkit	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		

程序包	AMI 最低版本	Container	最小容器
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-setools	4.4.1		
python3-setuptools	59.6.0		
python3-setuptools-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

程序包	AMI 最低版本	Container	最小容器
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
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		

程序包	AMI 最低版本	Container	最小容器
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		
util-linux	2.37.4		
util-linux-core	2.37.4		
vim-data	9.0.2153		
vim-minimal	9.0.2153		
which	2.21		
xfsprogs	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		

在 AWS Elastic Beanstalk 上的 AL2023

AWS Elastic Beanstalk 是一项用于部署和扩展 Web 应用程序和服务的服务。您只需上传代码，Elastic Beanstalk 会自动处理部署 — 从容量配置、负载平衡、自动扩展到应用程序运行状况监控。有关更多信息，请参阅 AWS Elastic Beanstalk。

要使用 Elastic Beanstalk，您需要创建一个应用程序，将应用程序版本以应用程序源包的形式（如 Java .war 文件）上传到 Elastic Beanstalk，然后提供一些有关该应用程序的信息。Elastic Beanstalk 会自动启动环境，然后创建并配置运行代码所需的 AWS 资源。有关更多信息，请参阅 AWS Elastic Beanstalk 开发人员指南。

Elastic Beanstalk Linux 平台使用 Amazon EC2 实例，这些实例运行 Amazon Linux。截至 2023 年 8 月 4 日，Elastic Beanstalk 提供以下基于 Amazon Linux 2023 的平台分支：Docker、Tomcat、Java SE、Node.js、PHP 和 Python。Elastic Beanstalk 正在努力向更多的 Elastic Beanstalk 平台发布对 AL2023 的支持。

有关 Elastic Beanstalk 平台支持和当前基于 AL2023 构建的平台的完整列表，可在 Elastic Beanstalk 开发人员指南中的 Elastic Beanstalk Linux 平台部分找到。

有关新的 Elastic Beanstalk 平台和现有平台版本的发布说明，可在 Elastic Beanstalk 发布说明中找到。

在 AWS CloudShell 中使用 AL2023

AWS CloudShell 是一个已经事先完成身份验证的浏览器式 Shell，您可以直接从 启动它。AWS 管理控制台您可以通过几种不同的方式从 AWS 管理控制台 导航到 CloudShell。有关更多信息，请参阅如何开始使用 AWS CloudShell？。

AWS CloudShell 目前基于 Amazon Linux 2，将迁移到 AL2023。从 2023 年 12 月 4 日开始，向 AL2023 的迁移将开始在所有 AWS 区域 推出。有关 CloudShell 迁移到 AL2023 的更多信息，请参阅 AWS CloudShell 从 Amazon Linux 2 迁移到 Amazon Linux 2023。

使用基于 AL2023 的 Amazon ECS AMI 托管容器化工作负载

> **Note**
>
> 有关如何在容器内使用 AL2023 的更多信息，请参阅 AL2023 在容器里。

Amazon Elastic Container Service (Amazon ECS) 是一种完全托管式的容器编排服务，可以帮助您轻松部署、管理和扩展容器化应用程序。作为一种完全托管式服务，Amazon ECS 内置有 AWS 配置和操作最佳实践。它与 AWS 和第三方工具如 Amazon Elastic Container Registry (Amazon ECR) 和 Docker 集成。这种集成使团队更容易专注于构建应用程序而不是环境。您可以在云端跨 AWS 区域运行和扩展容器工作负载，省去了管理控制面板的繁琐。

您可以使用基于 AL2023 的 Amazon ECS 优化 AMI 在 AL2023 上托管容器化工作负载。更多信息，请参阅 Amazon ECS-optimized AMI

与 AL2 相比，Amazon ECS 在 AL2023 中的变更

与 AL2 一样，AL2023 提供了作为 Amazon ECS Linux 实例运行所需的基础程序包。在 AL2 中，`containerd`、`docker` 和 `ecs-init` 程序包通过 `amazon-linux-extras` 提供，而 AL2023 将这些程序包包含在核心存储库中。

借助通过版本控制的存储库的确定性升级功能，默认情况下，每个 AL2023 AMI 都锁定到特定的存储库版本。AL2023 Amazon ECS 优化 AMI 也是如此。对环境的所有更新都可以在部署前进行仔细的管理和测试，并在出现问题时提供一种简单的方法来回退到先前 AMI 的内容。有关此 AL2023 功能的更多信息，请参阅通过版本控制的存储库进行确定性升级 AL2023。

AL2023 切换到了 cgroup v2，取代了 AL2 支持的 cgroup v1 接口。有关更多信息，请参阅 统一控制组层次结构 (cgroup v2)。

> **Note**
>
> 2023.2.20230920（首个 AL2023.2 发布版）之前的 AL2023 版本在 `systemd` 中存在一个错误，不能正确处理 cgroup 中的内存不足（OOM）问题。cgroup 中的所有进程总是被全部终止，而非按预期行为由 OOM-Killer 逐个选择进程终止。
> 与 AL2 的行为相比，这是一个功能回归，并已在 2023.2.20230920 版本的 AL2023 中修复。

构建 Amazon ECS 优化 AMI 的代码可在 amazon-ecs-ami GitHub 项目上找到。发布说明描述了哪个 AL2023 版本对应哪个 Amazon ECS AMI 版本。

自定义基于 AL2023 的 Amazon ECS 优化的 AMI

> **Important**
>
> 我们建议您使用 Amazon ECS 优化的 AL2023 AMI。更多信息，请参阅《Amazon Elastic Container Service 开发人员指南》中的 Amazon ECS 优化 AMI。

您可以使用 Amazon ECS 创建自定义 AMI 所用的同样的构建脚本。更多信息，请参阅 Amazon ECS 优化 Linux AMI 构建脚本。

在 AL2023 上使用 Amazon Elastic File System

Amazon Elastic File System (Amazon EFS) 提供无服务器的完全弹性文件存储，这使您无需预置或管理存储容量和性能即可共享文件数据。Amazon EFS 可在不中断应用程序的情况下按需扩展到 PB 级，并可在您添加和移除文件时自动扩涨或收缩。Amazon EFS 具有简单的 Web 服务界面，可让您快速方便地创建和配置文件系统。该服务为您管理所有文件存储基础设施，这意味着您可以避免部署、修补和维护复杂文件系统配置的复杂性。

Amazon EFS 支持 Network File System 版本 4（NFSv4.1 和 NFSv4.0）协议，因此，您当前使用的应用程序和工具可以与 Amazon EFS 无缝协作。多个计算实例（包括 Amazon EC2、Amazon ECS 和 AWS Lambda）可以同时访问一个 Amazon EFS 文件系统。因此，一个 EFS 文件系统可以为多个计算实例或服务器上运行的工作负载和应用程序提供通用数据源。

在 AL2023 上安装 **amazon-efs-utils**

`amazon-efs-utils` 程序包在 AL2023 存储库中可用，可安装并用于访问 Amazon EFS 文件系统。

在 AL2023 上安装 **amazon-efs-utils** 软件包

- 使用以下命令安装 `amazon-efs-utils`。

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

在 AL2023 上装载 Amazon EFS 文件系统

安装 `amazon-efs-utils` 后，您可以在 AL2023 实例上挂载 Amazon EFS 文件系统。

在 AL2023 上装载 Amazon EFS 文件系统

- 要使用文件系统 ID 进行挂载，请使用以下命令。

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

您也可以装载该文件系统，以便使用 TLS 或 DNS 名称或装载目标 IP 而非文件系统 id 来加密传输中的数据。有关更多信息，请参阅使用 EFS 装载帮助程序在 Amazon Linux 实例上装载文件系统。

使用基于 Amazon EMR AL2023

Amazon EMR 是一种 Web 服务，它使您可以轻松地利用 Apache Hadoop 和 AWS所提供的服务高效地处理海量数据。

AL2023 基于亚马逊 EMR 的版本

亚马逊 EMR 7.0.0 版本是第一个在此基础上构建的版本。 AL2023此版本 AL2023 是亚马逊 EMR 的基本操作系统，为亚马逊 EMR 带来了所有优点。 AL2023 更多信息，请参阅 Amazon EMR 7.0.0 发布说明。

AL2023 基于 EKS 的 Amazon EMR

EKS 6.13 上的 Amazon EMR 是第一个 AL2023 作为选项推出的版本。在此版本中，您可以将 Spark AL2023 作为操作系统与 Java 17 运行时一起启动。更多信息，请参阅 Amazon EMR on EKS 6.13 发布说明以及所有 Amazon EMR on EKS 发布说明。

AL2023 在中使用 AWS Lambda

使用 AWS Lambda，您无需预置或管理服务器即可运行代码。您只需为消耗的计算时间付费——代码不运行时不会产生费用。您可以运行几乎任何类型的应用程序或后端服务的代码——完全无需管理。只需上传您的代码，Lambda 就会处理以高可用性运行和扩展您的代码所需的一切。

AL2023 **`provided.al2023`**托管运行时和容器镜像

`provided.al2023`基本运行时基于AL2023 最小容器映像，并提供基 AL2023 于 Lambda 托管的运行时和容器基础映像。由于`provided.al2023`运行时间基于 AL2023 最小的容器映像，因此小于 40 MB 的运行时间要比大约 109 MB 的`provided.al2`运行时间小得多。

更多信息，请参阅 Lambda 运行时和使用 Lambda 容器映像。

AL2023 基于 Lambda 运行时

未来发布的托管语言运行时（例如 Node.js 20、Python 3.12、Java 21 和 .NET 8）基于基于 AL2023 运行时的公告中所述，并将`provided.al2023`用作基础映像。 AL2023

AL2023 基于 Lambda 函数

- AL2023 写入的 Lambda 函数 Go
- AL2023 写入的 Lambda 函数 Rust

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的 Lambda 运行时系统。

教程

以下教程向您展示如何使用运行 Amazon Linux 2023（AL2023）的 Amazon EC2 实例执行常见任务。有关视频教程，请参阅 AWS 教学视频和实验。

有关 AL2 的说明，请参阅《Amazon EC2 用户指南》中的运行 Linux 的 Amazon EC2 实例教程。

教程

- 教程：在上安装 LAMP 服务器 AL2023
- 教程： SSL/TLS 开启配置 AL2023
- 教程：在上发布 WordPress 博客 AL2023
- 教程：023 年 Redis 6 到 Valkey 过渡 AL2
- 教程：在上安装 GNOME 桌面环境 AL2023
- 教程：在上配置 TigerVNC 服务器 AL2023
- 在内核上使用多代 LRU (MGLRU) AL2023

教程：在上安装 LAMP 服务器 AL2023

以下过程可帮助您在 AL2023 您的实例（有时称为 LAMP 网络服务器或 LAMP 堆栈）上安装支持 PHP 和 M ar iaDB（社区开发的 MySQL 分支）的 Apache 网络服务器。您可以使用此服务器来托管静态网站或部署能对数据库中的信息执行读写操作的动态 PHP 应用程序。

> **Important**
>
> 这些程序旨在与一起使用 AL2023。如果您尝试在其他发行版（例如 Ubuntu 或红帽企业 Linux）上设置 LAMP Web 服务器，则本教程不适合。对于 Ubuntu，请参阅以下 Ubuntu 社区文档：SQLPHP。ApacheMy有关其他发布版本，请参阅特定于该版本的文档。

任务

- 步骤 1：准备 LAMP 服务器
- 步骤 2：测试 LAMP 服务器
- 步骤 3：确保数据库服务器的安全
- 步骤 4：（可选）安装 phpMyAdmin

- 故障排除
- 相关主题

步骤 1：准备 LAMP 服务器

先决条件

- 本教程假设您已经使用 AL2023一个可从互联网访问的公有 DNS 名称启动了一个新实例。有关更多信息，请参阅 Amazon EC2 上的 AL2023。您还必须配置安全组，以便允许 SSH (端口 22)、HTTP (端口 80) 和 HTTPS (端口 443) 连接。有关这些先决条件的更多信息，请参阅《Amazon EC2 用户指南》中的为您的 Linux 实例授权入站流量。
- 以下过程安装最新的 PHP 版本（当前为 AL2023 8.1）。如果您计划使用本教程中所述的 PHP 应用程序之外的 PHP 应用程序，则应检查其与 8.1 的兼容性。

准备 LAMP 服务器

1. 连接到您的 实例。有关更多信息，请参阅 连接到 AL2023 实例。
2. 为确保您的所有软件包都处于最新状态，请对您的实例执行快速软件更新。此过程可能需要几分钟的时间，但必须确保您拥有最新的安全更新和缺陷修复。

   -y 选项安装更新时不提示确认。如果您希望在安装前检查更新，则可以忽略该选项。

   ```
   [ec2-user ~]$ sudo dnf upgrade -y
   ```
3. 安装最新版本的 Apache Web 服务器和适用的 PHP 软件包。 AL2023

   ```
   [ec2-user ~]$ sudo dnf install -y httpd wget php-fpm php-mysqli php-json php php-devel
   ```
4. 安装 MariaDB 软件包。使用 dnf install 命令可同时安装多个软件包和所有相关依赖项。

   ```
   [ec2-user ~]$ sudo dnf install mariadb105-server
   ```

   您可以使用以下命令查看这些程序包的当前版本：

   ```
   [ec2-user ~]$ sudo dnf info package_name
   ```

   示例：

```
[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 (mariadbd) and many different client programs and
 libraries.
             : The base package contains the standard MariaDB/MySQL client programs
 and
             : utilities.
```

5. 启动 Apache Web 服务器。

```
[ec2-user ~]$ sudo systemctl start httpd
```

6. 使用 systemctl 命令配置 Apache Web 服务器，使其在每次系统启动时启动。

```
[ec2-user ~]$ sudo systemctl enable httpd
```

您可以通过运行以下命令验证 httpd 是否已启用：

```
[ec2-user ~]$ sudo systemctl is-enabled httpd
```

7. 如果您尚未这样做，请添加安全规则以允许与您的实例的入站 HTTP (端口 80) 连接。默认情况下，启动时会为您的实例创建启动向导***N***安全组。如果您没有添加其他安全组规则，此组仅包含允许 SSH 连接的一条规则。

a. 打开位于 https://console.aws.amazon.com/ec2/ 的 Amazon EC2 控制台。

b. 在左侧的导航中，选择 Instances（实例），然后选择您的实例。

c. 在安全选项卡上，查看入站规则。您应看到以下规则：

```
Port range   Protocol     Source
22           tcp          0.0.0.0/0
```

> **Warning**
>
> 使用`0.0.0.0/0`允许所有 IPv4 地址通过 SSH 访问您的实例。这在测试环境中可以接受一小段时间，但是在生产环境中并不安全。在生产环境中，您仅授权特定 IP 地址或地址范围访问您的实例。

d. 如果没有允许 HTTP（端口 80）连接的入站规则，则必须立即添加规则。选择安全组的链接。使用为您的 Linux 实例授权入站流量中的步骤，添加具有以下值的新入站安全规则：

- 类型：HTTP
- 协议：TCP
- Port Range：80
- Source：Custom

8. 测试您的 Web 服务器。在 Web 浏览器中，键入您的实例的公有 DNS 地址 (或公有 IP 地址)。如果 `/var/www/html` 中没有内容，您应该会看到 Apache 测试页面，该页面将显示消息 It works!（它工作正常！）。

   您可以使用 Amazon EC2 控制台获取实例的公有 DNS（查看公有 IPv4 DNS 列；如果此列处于隐藏状态，请选择首选项（齿轮形图标）并打开公 IPv4 有 DNS）。

   验证实例的安全组是否包含允许端口 80 上的 HTTP 流量的规则。更多信息，请参阅向安全组添加规则。

   > **Important**
   >
   > 如果您使用的不是 Amazon Linux，则还可能需要在实例上配置防火墙才能允许这些连接。有关如何配置防火墙的更多信息，请参阅适用于特定分配的文档。

Apache httpd 提供的文件保存在名为 Apache 文档根目录的目录中。Amazon Linux Apache 文档根目录为 /var/www/html，默认情况下归根用户所有。

要允许 ec2-user 账户操作此目录中的文件，必须修改其所有权和权限。有多种方式可以完成此任务。在本教程中，可将 ec2-user 添加到 apache 组，将 /var/www 目录的所有权授予 apache 组，并为该组指定写入权限。

设置文件权限

1. 将您的用户 (这里指 ec2-user) 添加到 apache。

```
[ec2-user ~]$ sudo usermod -a -G apache ec2-user
```

2. 先退出再重新登录以选取新组，然后验证您的成员资格。

   a. 退出（使用 exit 命令或关闭终端窗口）：

```
[ec2-user ~]$ exit
```

   b. 要验证您是否为 apache 组的成员，请重新连接到实例，然后运行以下命令：

```
[ec2-user ~]$ groups
ec2-user adm wheel apache systemd-journal
```

3. 将 /var/www 及其内容的组所有权更改到 apache 组。

```
[ec2-user ~]$ sudo chown -R ec2-user:apache /var/www
```

4. 要添加组写入权限以及设置未来子目录上的组 ID，请更改 /var/www 及其子目录的目录权限。

```
[ec2-user ~]$ sudo chmod 2775 /var/www && find /var/www -type d -exec sudo chmod
 2775 {} \;
```

5. 要添加组写入权限，请递归地更改 /var/www 及其子目录的文件权限：

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0664 {} \;
```

这样，ec2-user (和 apache 组的任何未来成员) 可以添加、删除和编辑 Apache 文档根目录中的文件，允许您添加内容，如静态网站或 PHP 应用程序。

保护您的 Web 服务器 (可选)

运行 HTTP 协议的 Web 服务器不为其发送或接收的数据提供传输安全。当您使用 Web 浏览器连接到 HTTP 服务器时，网络路径上任何地方的窃听者都可以看到您访问的内容、收到的网页内容以及您提交的任何 HTML 表单的内容（包括密码）。 URLs 保护您的 Web 服务器的最佳实践是安装 HTTPS (HTTP Secure) 支持，它将使用 SSL/TLS 加密保护您的数据。

有关在服务器上启用 HTTPS 的信息，请参阅 教程： SSL/TLS 开启配置 AL2023。

步骤 2：测试 LAMP 服务器

如果服务器已安装并运行，且文件权限设置正确，则 `ec2-user` 账户应该能够在 `/var/www/html` 目录 (可从 Internet 访问) 中创建 PHP 文件。

测试您的 LAMP 服务器

1. 在 Apache 文档根目录中创建一个 PHP 文件。

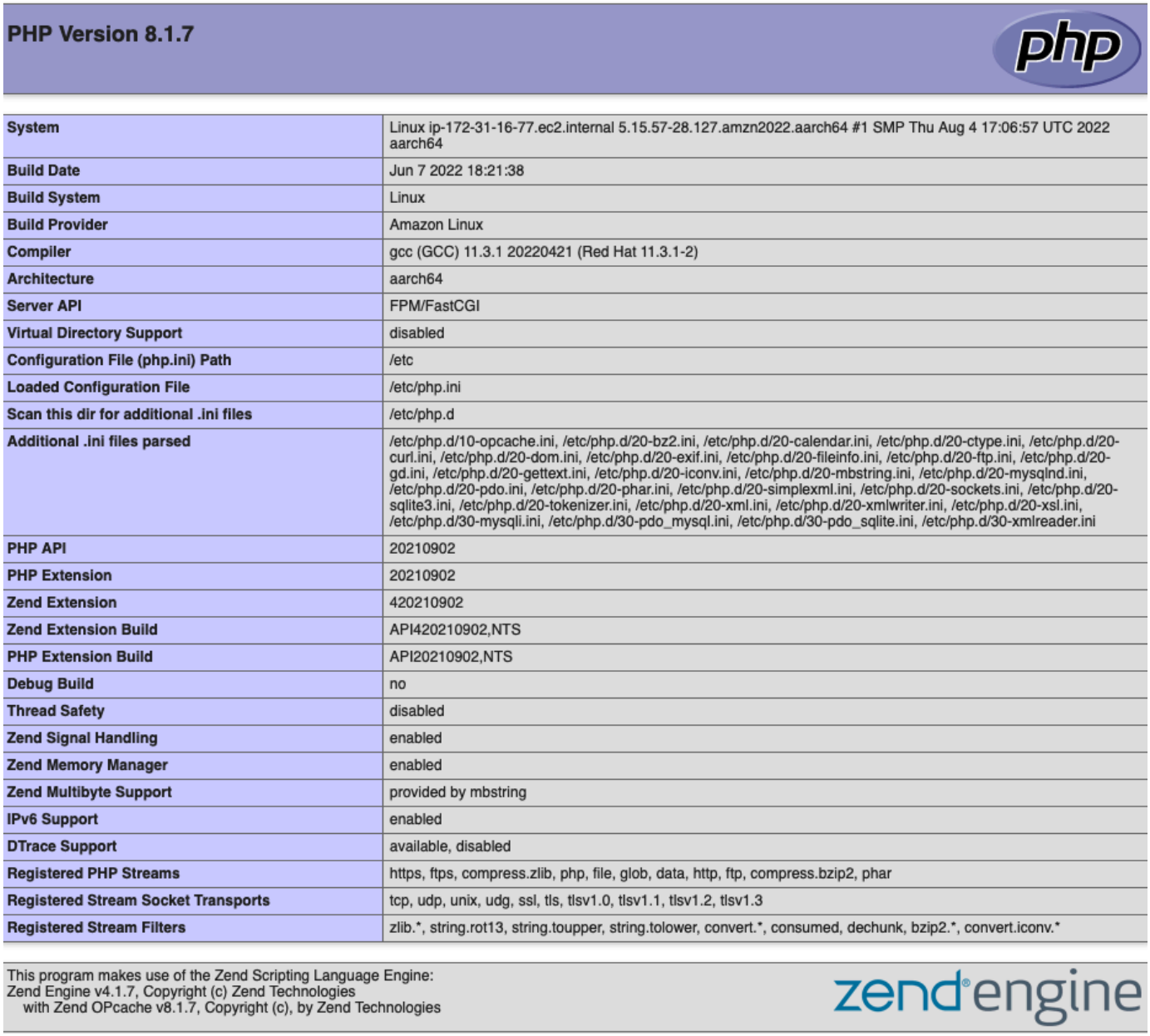
   ```
   [ec2-user ~]$ echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
   ```

   尝试运行该命令时，如果出现“Permission denied (权限被拒绝)”错误，请尝试先注销，再重新登录，以获取您在 设置文件权限 中配置的适当组权限。

2. 在 Web 浏览器中，键入您刚刚创建的文件的 URL。此 URL 是实例的公用 DNS 地址，后接正斜杠和文件名。例如：

   ```
   http://my.public.dns.amazonaws.com/phpinfo.php
   ```

   您应该会看到 PHP 信息页面：



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-xmlreader.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

如果您未看到此页面，请验证上一步中是否已正确创建 `/var/www/html/phpinfo.php` 文件。您还可以使用以下命令验证已经安装了所有必需的程序包。

```
[ec2-user ~]$ sudo dnf list installed httpd mariadb-server php-mysqlnd
```

如果输出中未列出任何必需的程序包，请使用 sudo yum install ***package*** 命令安装它们。

3. 删除 `phpinfo.php` 文件。尽管此信息可能很有用，但出于安全考虑，不应将其传播到 Internet。

```
[ec2-user ~]$ rm /var/www/html/phpinfo.php
```

现在，您应该有了一个功能完善的 LAMP Web 服务器。如果您将内容添加到 Apache 文档根目录 (位于 /var/www/html)，您应该能够在您的实例的公有 DNS 地址中看到该内容。

步骤 3：确保数据库服务器的安全

MariaDB 服务器的默认安装提供有多种功能，这些功能对于测试和开发都很有帮助，但对于产品服务器，应禁用或删除这些功能。mysql_secure_installation 命令可引导您设置根密码并删除安装中的不安全功能。即使您不打算使用 MariaDB 服务器，我们也建议执行此步骤。

保护 MariaDB 服务器

1. 启动 MariaDB 服务器。

```
[ec2-user ~]$ sudo systemctl start mariadb
```

2. 运行 mysql_secure_installation。

```
[ec2-user ~]$ sudo mysql_secure_installation
```

   a. 在提示时，键入根账户的密码。

      i. 键入当前根密码。默认情况下，根账户没有设置密码。按 Enter。

      ii. 键入 **Y** 设置密码，然后键入两次安全密码。有关创建安全密码的更多信息，请参阅https://identitysafe.norton.com/password-generator/。确保将此密码存储在安全位置。

      设置 MariaDB 根密码仅是保护数据库的最基本措施。在您构建或安装数据库驱动的应用程序时，您通常可以为该应用程序创建数据库服务用户，并避免使用根账户执行除数据库管理以外的操作。

   b. 键入 **Y** 删除匿名用户账户。

   c. 键入 **Y** 禁用远程根登录。

   d. 键入 **Y** 删除测试数据库。

   e. 键入 **Y** 重新加载权限表并保存您的更改。

3. (可选) 如果您不打算立即使用 MariaDB 服务器，请停止它。您可以在需要时再次重新启动。

```
[ec2-user ~]$ sudo systemctl stop mariadb
```

4. (可选) 如果您希望每次启动时 MariaDB 服务器都启动，请键入以下命令。

```
[ec2-user ~]$ sudo systemctl enable mariadb
```

步骤 4：（可选）安装 phpMyAdmin

phpMyAdmin是一款基于 Web 的数据库管理工具，可用于查看和编辑 EC2 实例上的 MySQL 数据库。按照下述步骤操作，在您的 Amazon Linux 实例上安装和配置 phpMyAdmin。

> **Important**
>
> 除非您已在 Apache SSL/TLS 中启用，否则我们不建议使用phpMyAdmin来访问 LAMP 服务器；否则，您的数据库管理员密码和其他数据将不安全地通过 Internet 传输。有关开发人员提出的安全建议，请参阅保护您的 phpMyAdmin 安装。有关在 EC2 实例上保护 Web 服务器的一般信息，请参阅 教程： SSL/TLS 开启配置 AL2023。

要安装 phpMyAdmin

1. 安装所需的依赖项。

   ```
   [ec2-user ~]$ sudo dnf install php-mbstring php-xml -y
   ```

2. 重启 Apache。

   ```
   [ec2-user ~]$ sudo systemctl restart httpd
   ```

3. 重启 php-fpm。

   ```
   [ec2-user ~]$ sudo systemctl restart php-fpm
   ```

4. 导航到位于 /var/www/html 的 Apache 文档根。

   ```
   [ec2-user ~]$ cd /var/www/html
   ```

5. 从 https://www.phpmyadmin.net/downloads 选择最新 phpMyAdmin 版本的源软件包。要将文件直接下载到您的实例，请复制链接并将其粘贴到 wget 命令，如本示例中所述：

   ```
   [ec2-user html]$ wget https://www.phpmyadmin.net/downloads/phpMyAdmin-latest-all-languages.tar.gz
   ```

6. 使用以下命令创建 phpMyAdmin 文件夹并将程序包提取到其中。

```
[ec2-user html]$ mkdir phpMyAdmin && tar -xvzf phpMyAdmin-latest-all-languages.tar.gz -C phpMyAdmin --strip-components 1
```

7. 删除*phpMyAdmin-latest-all-languages.tar.gz*压缩包。

```
[ec2-user html]$ rm phpMyAdmin-latest-all-languages.tar.gz
```

8. (可选) 如果 MySQL 服务器未运行，请立即启动它。

```
[ec2-user ~]$ sudo systemctl start mariadb
```

9. 在 Web 浏览器中，键入 phpMyAdmin 安装的 URL。此 URL 是实例的公有 DNS 地址 (或公有 IP 地址)，后接正斜杠和您安装目录的名称。例如：

```
http://my.public.dns.amazonaws.com/phpMyAdmin
```

   你应该会看到 phpMyAdmin 登录页面：

10. 使用您之前创建的`root`用户名和 MySQL 根密码登录您的 phpMyAdmin 安装。

    您的安装仍需进行配置，然后才能投入使用。我们建议您首先手动创建配置文件，如下所示：

    a. 要从最小的配置文件开始，请使用您常用的文本编辑器创建一个新文件，然后将 `config.sample.inc.php` 的内容复制到该文件中。

    b. 将文件另存为`config.inc.php`包含的 phpMyAdmin 目录中`index.php`。

    c. 有关任何其他设置，请参阅安装说明的 “使用 phpMyAdmin 安装脚本” 部分中的文件创建后说明。

    有关使用的信息 phpMyAdmin，请参阅《phpMyAdmin 用户指南》。

故障排除

本部分提供了解决在设置新 LAMP 服务器时可能遇到的常见问题的建议。

我无法使用 Web 浏览器连接到我的服务器

执行以下检查以查看您的 Apache Web 服务器是否正在运行且可以访问。

- Web 服务器正在运行吗？

  您可以通过运行以下命令验证 httpd 是否已启用：

  ```
  [ec2-user ~]$ sudo systemctl is-enabled httpd
  ```

  如果 httpd 进程未运行，请重复准备 LAMP 服务器中描述的步骤。

- 防火墙是否配置正确？

  验证实例的安全组是否包含允许端口 80 上的 HTTP 流量的规则。更多信息，请参阅向安全组添加规则。

我无法使用 HTTPS 连接到我的服务器

执行以下检查以查看 Apache Web 服务器是否配置为支持 HTTPS。

- Web 服务器配置是否正确？

  安装 Apache 后，服务器将针对 HTTP 流量进行配置。要支持 HTTPS，请在服务器上启用 TLS 并安装 SSL 证书。有关信息，请参阅 教程： SSL/TLS 开启配置 AL2023。

- 防火墙是否配置正确？

  验证实例的安全组是否包含允许端口 443 上的 HTTPS 流量的规则。更多信息，请参阅为您的 Linux 实例授权入站流量。

相关主题

有关将文件传输到您的实例或在 Web 服务器上安装 WordPress 博客的更多信息，请参阅以下文档：

- 《Amazon EC2 用户指南》中的使用 WinSCP 将文件传输到 Linux 实例。

- 《Amazon EC2 用户指南》中的使用 SCP 客户端将文件传输到 Linux 实例。
- 教程：在上发布 WordPress 博客 AL2023

有关本教程中使用的命令和软件的更多信息，请参阅以下网页：

- Apache Web 服务器：http://httpd.apache.org/
- MariaDB 数据库服务器：https://mariadb.org/
- PHP 编程语言：http://php.net/

有关注册 Web 服务器域名或将现有域名转移到此主机的更多信息，请参阅 Amazon Route 53 开发人员指南中的创建域和子域并将其迁移到 Amazon Route 53。

教程： SSL/TLS 开启配置 AL2023

安全套接字 Layer/Transport 层安全（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在带有 AL2023 Apache Web 服务器的 EC2 实例上。本教程假定您未使用负载均衡器。如果您正在使用 Elastic Load Balancing，则可以选择使用来自 AWS Certificate Manager 的证书在负载均衡器上配置 SSL 卸载。

由于历史原因，Web 加密通常简称为 SSL。虽然 Web 浏览器仍支持 TLS，但下一代协议 TLS 不容易受到攻击。默认情况下， AL2023 为所有版本的 SSL 禁用服务器端支持。安全标准机构认为 TLS 1.0 不安全。TLS 1.0 和 TLS 1.1 已于 2021 年 3 月正式弃用。本教程仅包含有关启用 TLS 1.2 的指导。TLS 1.3 已于 2018 年完成， AL2 只要支持和启用底层 TLS 库（本教程中的 OpenSSL），即可使用。客户端必须在 2023 年 6 月 28 日之前支持 TLS 1.2 或更高版本。有关更新的加密标准的更多信息，请参阅 RFC 7568 和 RFC 8446。

在本教程中，将现代 Web 加密简称为 TLS。

> **Important**
>
> 这些步骤适用于 AL2023。如果您尝试设置运行不同分配的 EC2 实例，或者运行旧版本 Amazon Linux 的实例，则本教程中的一些过程可能不适合。对于 Ubuntu，请参阅以下 Ubuntu 社区文档：Open SSL on Ubuntu。有关 Red Hat Enterprise Linux 的信息，请参阅以下：设置 Apache HTTP Web 服务器。有关其他发布版本，请参阅特定于该版本的文档。

> **Note**
>
> 或者，您可以将 AWS Certificate Manager (ACM) 用于 AWS Nitro 安全区，这是一种安全区应用程序，允许您在带有 Nitro Enclaves 的 Amazon EC2 实例上运行的 Web 应用程序和服务器上使用公有和私有 SSL/TLS 证书。 AWS Nitro Enclaves 是一项 Amazon EC2 功能，它允许创建隔离的计算环境，以保护和安全地处理高度敏感的数据，例如 SSL/TLS 证书和私钥。适用于 Nitro Enclaves 的 ACM 与运行在 Amazon EC2 Linux 实例上的 nginx 结合使用，以创建私有密钥、分发证书和私有密钥以及管理证书续订。
> 要使用适用于 Nitro Enclaves 的 ACM，必须使用启用了 Enclave 的 Linux 实例。
> 有关更多信息，请参阅什么是 AWS 硝基飞地？ 以及AWS Certificate Manager 《硝基飞地用户指南》AWS 中的 Nitro Enclaves。

内容

前提条件

在开始本教程之前，请完成以下步骤：

- 启动 EBS 支持的实例 AL2023 。有关更多信息，请参阅 Amazon EC2 上的 AL2023。
- 配置安全组以允许您的实例接受以下 TCP 端口上的连接：
  - SSH (端口 22)
  - HTTP (端口 80)
  - HTTPS (端口 443)

  有关更多信息，请参阅《Amazon EC2 用户指南》中的为您的 Linux 实例授权入站流量。
- 安装 Apache Web 服务器。有关 step-by-step说明，请参阅教程：在上安装 LAMP 服务器 AL2023。仅需要 httpd 包及其依赖项，因此可以忽略涉及 PHP 和 MariaDB 的说明。
- 要识别和验证网站，TLS 公有密钥基础设施 (PKI) 依赖于域名系统 (DNS)。要使用 EC2 实例托管公共网站，您需要为 Web 服务器注册一个域名，或者将现有域名转让给您的 Amazon EC2 主机。

可通过很多第三方域注册和 DNS 托管服务来执行此操作，也可以使用 Amazon Route 53 执行此操作。

步骤 1：在服务器上启用 TLS

此过程将引导您完成 AL2023 使用自签名数字证书设置 TLS 的过程。

> **Note**
>
> 自签名证书对于测试是可接受的，但对于生产不是。如果您将自签名证书公开到 Internet，您网站的访客将会看到安全警告。

在服务器上启用 TLS

1. 连接到您的实例并确认 Apache 正在运行。有关更多信息，请参阅 连接到 AL2023 实例。

   ```
   [ec2-user ~]$ sudo systemctl is-enabled httpd
   ```

   如果返回的值不是“启用”，则启动 Apache 并将它设置为每次随系统一起启动。

   ```
   [ec2-user ~]$ sudo systemctl start httpd && sudo systemctl enable httpd
   ```

2. 为确保您的所有软件包都处于最新状态，请对您的实例执行快速软件更新。此过程可能需要几分钟的时间，但必须确保您拥有最新的安全更新和缺陷修复。

   > **Note**
   >
   > -y 选项安装更新时不提示确认。如果您希望在安装前检查更新，则可以忽略该选项。

   ```
   [ec2-user ~]$ sudo dnf install openssl mod_ssl
   ```

3. 输入以下命令后，系统将向您显示提示框，您可以在其中输入有关站点的信息。

   ```
   [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
   ```

这将生成两个文件：

- /etc/pki/tls/private/apache-selfsigned.key（私钥）
- /etc/pki/tls/certs/apache-selfsigned.crt（自签名证书）

证书文件名与中SSLCertificateFile指令中分配的默认名称相匹配/etc/httpd/conf.d/ssl.conf。

您的实例现在具有以下文件，可使用这些文件配置安全服务器并创建证书以进行测试：

- /etc/httpd/conf.d/ssl.conf

  mod_ssl 的配置文件。它包含一些指令 以指示 Apache 在何处查找以下信息：加密密钥和证书、要允许的 TLS 协议版本以及要接受的加密密码。这将是您的本地证书文件：

- /etc/pki/tls/certs/apache-selfsigned.crt
- /etc/pki/tls/private/apache-selfsigned.key

该.crt文件包含自签名证书，.key文件包含私钥。Apache 要求证书和密钥采用 PEM 格式，其中包含 Base64 编码的 ASCII 字符，并用“BEGIN”和“END”行框起来，如以下简短示例所示。

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQD2KKx/8Zk94m1q
3gQMZF9ZN66Ls19+3tHAgQ5Fpo9KJDhzLjOOCI8u1PTcGmAah5kEitCEc0wzmNeo
BCl0wYR6G0rGaKtK9Dn7CuIjvubtUysVyQoMVPQ97ldeakHWeRMiEJFXg6kZZ0vr
GvwnKoMh3DlK44D9dX7IDua2PlYx5+eroA+1Lqf32ZSaAO0bBIMIYTHigwbHMZoT
...
56tE7THvH7vOEf4/iUOsIrEzaMaJ0mqkmY1A70qQGQKBgBF3H1qNRNHuyMcPODFs
27hDzPDinrquSEvoZIggkDMlh2irTiipJ/GhkvTpoQlv0fK/VXw8vSgeaBuhwJvS
LXU9HvYq0U6O4FgD3nAyB9hI0BE13r1HjUvbjT7moH+RhnNz6eqqdscCS09VtRAo
4QQvAqOa8UheYeoXLdWcHaLP
-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----
MIIEazCCA1OgAwIBAgICWxQwDQYJKoZIhvcNAQELBQAwgbExCzAJBgNVBAYTAi0t
MRIwEAYDVQQIDAlTb21lU3RhdGUxETAPBgNVBAcMCFNvbWVDaXR5MRkwFwYDVQQK
DBBTb21lT3JnYW5pemF0aW9uMR8wHQYDVQQLDBZTb21lT3JnYW5pemF0aW9uYWxV
bml0MRkwFwYDVQQDDBBpcC0xNzItMzEtMjAtMjM2MSQwIgYJKoZIhvcNAQkBFhVy
...
z5rRUE/XzxRLBZOoWZpNWTXJkQ3uFYH6s/sBwtHpKKZMzOvDedREjNKAvk4ws6F0
```

```
CuIjvubtUysVyQoMVPQ97ldeakHWeRMiEJFXg6kZZ0vrGvwnKoMh3DlK44D9dlU3
WanXWehT6FiSZvB4sTEXXJN2jdw8g+sHGnZ8zCOsclknYhHrCVD2vnBlZJKSZvak
3ZazhBxtQSukFMOnWPP2a0DMMFGYUHOd0BQE8sBJxg==
-----END CERTIFICATE-----
```

文件名和扩展名只是为了提供便利，对功能没有影响。例如，只要 `ssl.conf` 文件中的相关指令使用相同的名称，您就可以将证书命名为 `cert.crt`、`cert.pem` 或任何其他文件名。

> **Note**
>
> 在使用您自己的自定义文件替换默认 TLS 文件时，请确保它们采用 PEM 格式。

4. 重启 Apache。

```
[ec2-user ~]$ sudo systemctl restart httpd
```

> **Note**
>
> 确保 TCP 端口 443 在您的 EC2 实例上是可访问的，如之前所述。

5. 现在，您的 Apache Web 服务器应通过端口 443 支持 HTTPS (安全 HTTP)。通过将您的 EC2 实例的 IP 地址或完全限定域名与前缀 **`https://`** 一起输入浏览器 URL 栏中来对其进行测试。

   由于您正在使用自签名的不可信主机证书连接到站点，因此您的浏览器可能会显示一系列安全警告。忽视这些警告并继续连接站点。

   如果默认 Apache 测试页面打开，这意味着您已成功在服务器上配置 TLS。在浏览器和服务器之间传输的所有数据现在都已加密。

> **Note**
>
> 为了防止站点访问者遇到警告屏幕，您必须获取一个可信 CA 签名证书，该证书不仅进行加密，而且还公开验证您是否为站点拥有者。

步骤 2：获取 CA 签名的证书

您可以使用以下过程获取 CA 签名证书：

- 从私有密钥生成证书签名请求 (CSR)
- 将 CSR 提交给证书颁发机构 (CA)
- 获取签名的主机证书
- 配置 Apache 以使用证书

从加密角度看，自签名 TLS X.509 主机证书与 CA 签名证书完全相同。二者之间的区别在于社交层面，而非数学层面。CA 承诺，在向申请者颁发证书之前，至少验证域的所有权。每个 Web 浏览器都包含浏览器供应商 CAs 信任的列表。X.509 证书主要包含一个与您的私有服务器密钥对应的公有密钥和一个以加密方式与该公有密钥关联的 CA 的签名。当浏览器通过 HTTPS 连接到 Web 服务器时，服务器会提供证书供浏览器根据其可信列表进行检查 CAs。如果签署人位于列表上，或可通过由其他可信签署人组成的一系列信任访问，则浏览器将与服务器协商一个快速加密数据通道并加载页面。

由于验证请求需要投入人力，证书通常会产生费用，因此应货比三家。一些公司免费 CAs 提供基本级别的证书。其中最引人注目的是 CAs L et's Encrypt 项目，该项目还支持证书创建和续订过程的自动化。有关使用 Let's Encrypt 证书的更多信息，请参阅获取 Certbot。

如果您打算提供商业级服务，AWS Certificate Manager 是一个不错的选择。

主机证书的基础是密钥。从 2019 年开始，政府和行业群体建议 RSA 密钥使用 2048 位的最小密钥（模数）大小，旨在将文档一直保护到 2030 年。OpenSSL 在中生成的默认模数大小为 2048 位，适用于 AL2023 在 CA 签名的证书中使用。在以下过程中，为需要自定义密钥的人员提供了一个可选步骤，例如，具有较大模数或使用不同加密算法的步骤。

> **Important**
>
> 除非您拥有注册并托管的 DNS 域，否则，有关获取 CA 签名主机证书的这些说明不适用。

获取 CA 签名的证书

1. 连接到您的实例并导航到etc/pki/tls/private//。这是存储 TLS 的服务器私有密钥的目录。如果您希望使用现有的主机密钥生成 CSR，请跳到步骤 3。有关连接到实例的更多信息，请参阅 连接到 AL2023 实例
2. (可选) 生成新的私有密钥。下面是一些密钥配置示例。任何生成的密钥都可用于您的 Web 服务器，但它们实施安全的程度和类型有所不同。

   - 示例 1：创建默认 RSA 主机密钥。生成的文件 **`custom.key`** 是一个 2048 位 RSA 私有密钥。

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key
```

- 示例 2：创建具有更大模数的更严格的 RSA 密钥。生成的文件 **custom.key** 是一个 4096 位 RSA 私有密钥。

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key 4096
```

- 示例 3：创建具有密码保护的 4096 位加密的 RSA 密钥。生成的文件 **custom.key** 是一个已使用 AES-128 密码加密的 4096 位 RSA 私有密钥。

> **Important**
>
> 对密钥进行加密可增强安全性，但由于加密的密钥需要密码，因此依赖于加密密钥的服务无法自动启动。每当您使用此密钥时，都必须通过 SSH 连接提供密码（在上一示例中为“abcde12345”）。

```
[ec2-user ~]$ sudo openssl genrsa -aes128 -passout pass:abcde12345 -out
 custom.key 4096
```

- 示例 4：使用非 RSA 密码创建密钥。RSA 加密可能相对较慢，因为其公有密钥的大小基于两个大素数的乘积。不过，可以为 TLS 创建使用非 RSA 密码的密钥。在交付同等级别的安全性时，基于椭圆曲线的数学运算的密钥更小，计算起来更快。

```
[ec2-user ~]$ sudo openssl ecparam -name prime256v1 -out custom.key -genkey
```

结果为一个使用 prime256v1（OpenSSL 支持的“命名曲线”）的 256 位椭圆曲线私有密钥。根据 NIST，其加密强度略高于 2048 位 RSA 密钥。

> **Note**
>
> 并非所有密钥都 CAs 提供与 RSA elliptic-curve-based 密钥相同的支持级别。

确保新的私钥具有高度限制的所有权和权限（owner=root，group=root，仅适用于所有者）。read/write 命令将如以下示例所示。

```
[ec2-user ~]$ sudo chown root:root custom.key
```

```
[ec2-user ~]$ sudo chmod 600 custom.key
[ec2-user ~]$ ls -al custom.key
```

上述命令生成以下结果。

```
-rw------- root root custom.key
```

在创建并配置满意的密钥后，可以创建 CSR。

3. 使用您首选的密钥创建 CSR。下面的示例使用了 **custom.key**。

```
[ec2-user ~]$ sudo openssl req -new -key custom.key -out csr.pem
```

OpenSSL 将打开一个对话框，并提示您输入下表中显示的信息。对于基本的经域验证的主机证书来说，除 Common Name 以外的所有字段都是可选字段。

Name	说明	示例
国家/地区名称	代表国家/地区的两个字母 ISO 缩写。	US (=美国)
州或省名称	组织所在州或省的名称。此名称不可使用缩写。	Washington
所在地名称	您的组织所在的位置，例如城市。	Seattle
组织名称	组织的法定全称。请勿缩写组织名称。	Example Corporation
组织部门名称	额外的组织信息 (如果有)。	示例部门
公用名	此值必须与您希望用户输入浏览器中的 Web 地址完全匹配。通常，这表示以主机名称为前缀的域名或采用 **www.example.com** 格式的别名。在使用自签名证书且没有 DNS 解析进行测试时，公用名可能仅包含主机名。 CAs 还提供更昂贵的证书，这些证书接受通配符名称，例如. ***.example.com**	www.example.com

Name	说明	示例
电子邮件地址	服务器管理员的电子邮件地址。	someone@example.com

最后，OpenSSL 将提示您输入可选的质询密码。此密码仅适用于 CSR 和您与 CA 之间的事务，因此请遵循 CA 提供的有关此密码以及其他可选字段、可选公司名的建议。CSR 质询密码不会影响服务器操作。

生成的文件 **csr.pem** 包含您的公有密钥、您的公有密钥的数字签名以及您输入的元数据。

4. 将 CSR 提交给 CA。这通常包括在文本编辑器中打开 CSR 文件并将内容复制到 Web 表格中。此时，可能会要求您提供一个或多个主题备用名称 (SANs) 以放在证书上。如果 **www.example.com** 是公用名，则 **example.com** 将是一个很好的 SAN，反之亦然。您网站的访客如果输入这两个名称的任何一个，便可看到一个没有错误的连接。如果您的 CA Web 表单允许，请在列表中包含常用名 SANs。有些是自动 CAs 包含的。

   在您的请求获得批准后，您将收到一个由 CA 签署的新主机证书。此外，系统可能会指示您下载中间证书文件，该文件包含完成 CA 的信任链所需的其他证书。

   > **Note**
   >
   > 您的 CA 可能会针对各种用途发送多种格式的文件。在本教程中，您应只使用 PEM 格式的证书文件，此格式通常会（但不总是）标有 `.pem` 或 `.crt` 文件扩展名。如果您不确定要使用哪个文件，请使用文本编辑器打开这些文件，并查找一个包含一个或多个以下面的行开始的块的文件。
   >
   > ```
   > - - - - -BEGIN CERTIFICATE - - - - -
   > ```
   >
   > 该文件还应以下面的行结束。
   >
   > ```
   > - - - -END CERTIFICATE - - - - -
   > ```
   >
   > 您还可以在命令行上测试文件，如下所示。
   >
   > ```
   > [ec2-user certs]$ openssl x509 -in certificate.crt -text
   > ```

验证这些行是否显示在文件中。请勿使用结尾为 .p7b、.p7c 或类似文件扩展名的文件。

5. 将新的 CA 签名证书和任何中间证书放在 /etc/pki/tls/certs 目录中。

> **Note**
>
> 可通过多种方法将新证书上传到 EC2 实例，但最直接、最有益的方法是在本地计算机和 EC2 实例上打开一个文本编辑器（例如，vi、nano 或记事本），然后在这两者之间复制并粘贴文件内容。在 EC2 实例上执行这些操作时，您需要根 [sudo] 权限。这样，一旦有任何权限或路径问题，您可以立即看到。但请小心操作，不要在复制内容时添加任何多余的行或以任何方式更改内容。

在/etc/pki/tls/certs目录内部，检查文件所有权、组和权限设置是否与高度严格的 AL2023 默认值相匹配（owner=root，group=root，仅适用于所有者）。 read/write 以下示例显示了要使用的命令。

```
[ec2-user certs]$ sudo chown root:root custom.crt
[ec2-user certs]$ sudo chmod 600 custom.crt
[ec2-user certs]$ ls -al custom.crt
```

这些命令应生成以下结果。

```
-rw------- root root custom.crt
```

中间证书文件的权限并不严格 (所有者=根、组=根、所有者可以写入、组可以读取、任何人均可读取)。以下示例显示了要使用的命令。

```
[ec2-user certs]$ sudo chown root:root intermediate.crt
[ec2-user certs]$ sudo chmod 644 intermediate.crt
[ec2-user certs]$ ls -al intermediate.crt
```

这些命令应生成以下结果。

```
-rw-r--r-- root root intermediate.crt
```

6. 将用于创建 CSR 的私有密钥放在 /etc/pki/tls/private/ 目录中。

> **Note**
>
> 可通过多种方法将自定义密钥上传到 EC2 实例，但最直接、最有益的方法是在本地计算机和 EC2 实例上打开一个文本编辑器（例如，vi、nano 或记事本），然后在这两者之间复制并粘贴文件内容。在 EC2 实例上执行这些操作时，您需要根 [sudo] 权限。这样，一旦有任何权限或路径问题，您可以立即看到。但请小心操作，不要在复制内容时添加任何多余的行或以任何方式更改内容。

在/etc/pki/tls/private目录内部，使用以下命令验证文件所有权、组和权限设置是否与高度严格的 AL2023 默认值相匹配（owner=root，group=root，仅适用于所有者）。 read/write

```
[ec2-user private]$ sudo chown root:root custom.key
[ec2-user private]$ sudo chmod 600 custom.key
[ec2-user private]$ ls -al custom.key
```

这些命令应生成以下结果。

```
-rw------- root root custom.key
```

7. 编辑 /etc/httpd/conf.d/ssl.conf 以反映您的新证书和密钥文件。

> **Note**
>
> 确保SSLEngine on在你的 VirtualHost 方块中设置了它。

a. 在 Apache 的 SSLCertificateFile 指令中提供 CA 签名主机证书的路径和文件名：

```
SSLCertificateFile /etc/pki/tls/certs/custom.crt
```

b. 如果您收到一个中间证书文件（此示例中为 intermediate.crt），请使用 Apache 的 SSLCACertificateFile 指令提供其路径和文件名：

```
SSLCACertificateFile /etc/pki/tls/certs/intermediate.crt
```

> **Note**
>
> 有些证书将主机证书和中间证书 CAs 合并到一个文件中，因此没有必要使用该SSLCACertificateFile指令。请查询您的 CA 提供的说明。

c. 在 Apache 的 SSLCertificateKeyFile 指令中提供私有密钥的路径和文件名（在该示例中为 custom.key）：

```
SSLCertificateKeyFile /etc/pki/tls/private/custom.key
```

8. 保存 /etc/httpd/conf.d/ssl.conf 并重启 Apache。

```
[ec2-user ~]$ sudo systemctl restart httpd
```

9. 通过在浏览器 URL 栏中输入带有 https:// 前缀的域名来测试您的服务器。您的浏览器应通过 HTTPS 加载测试页面而不会产生错误。

步骤 3：测试和强化安全配置

在 TLS 可操作且公开发布后，应测试其实际安全性。使用在线服务（例如 Qualys SSL Labs，该服务可对您的安全设置执行免费的全面分析）可轻松执行此操作。根据结果，您可以决定通过控制接受的协议、首选的密码和排除的密码来强化默认安全配置。有关更多信息，请参阅 Qualys 如何用公式表示其分数。

> **Important**
>
> 实际测试对服务器的安全性非常重要。少量配置错误可能导致严重的安全漏洞和数据丢失。由于建议的安全实践会不断变化以响应调查和新兴威胁，因此定期安全审核对于良好的服务器管理来说是必不可少的。

在 Qualys SSL Labs 站点上，使用 **www.example.com** 格式输入服务器的完全限定域名。约两分钟后，您将收到您站点的评级 (从 A 到 F) 和结果的详细信息。下表汇总了设置与默认 Apache 配置相同且具有默认 Cert AL2023 bot 证书的域的报告。

总评	B

证书	100%
协议支持	95%
密钥交换	70%
密码强度	90%

虽然概述信息显示配置基本正确，但详细报告标记了几个潜在的问题（在此处按严重性顺序列出）：

✗ 某些较旧的浏览器支持使用该 RC4 密码。密码是加密算法的数学核心。 RC4是一种用于加密 TLS 数据流的快速密码，众所周知，它有几个严重的弱点。除非您有充分理由支持旧版浏览器，否则，应禁用该密码。

✗ 支持旧 TLS 版本。该配置支持 TLS 1.0（已弃用）和 TLS 1.1（即将弃用）。从 2018 年开始，仅建议使用 TLS 1.2。

✗ 不完全支持向前保密性。向前保密性是一种算法功能，它使用从私有密钥派生的临时会话密钥进行加密。这意味着，在实践中，攻击者无法解密 HTTPS 数据，即使他们拥有 Web 服务器的长期私有密钥。

纠正 TLS 配置并供将来使用

1. 在文本编辑器中打开 `/etc/httpd/conf.d/ssl.conf` 配置文件，并在以下行的开头输入“#”以注释掉该行。

   ```
   #SSLProtocol all -SSLv3
   ```

2. 添加以下指令：

   ```
   #SSLProtocol all -SSLv3
   SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
   ```

   该指令显式禁用 SSL 版本 2 和 3 以及 TLS 版本 1.0 和 1.1。现在，服务器拒绝接受与使用 TLS 1.2 以外的任何协议的客户端之间的加密连接。指令中的冗长文字更清楚地向人类读者阐述为服务器配置的用途。

> **Note**
>
> 以此方式禁用 TLS 1.0 和 1.1 版可阻止一小部分过时的 Web 浏览器访问您的网站。

修改允许的密码列表

1. 在 `/etc/httpd/conf.d/ssl.conf` 配置文件中，找到包含 **`SSLCipherSuite`** 指令的部分，并通过在现有行的开头输入“#”来注释掉该行。

```
#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

2. 指定显式的密码套件，并指定密码顺序以优先使用向前保密性并避免不安全的密码。此处使用的 `SSLCipherSuite` 指令基于 Mozilla SSL 配置生成器的输出，该生成器根据服务器上运行的特定软件定制 TLS 配置。首先，通过使用以下命令的输出确定 Apache 和 OpenSSL 版本。

```
[ec2-user ~]$ yum list installed | grep httpd

[ec2-user ~]$ yum list installed | grep openssl
```

例如，如果返回的信息是 Apache 2.4.34 和 OpenSSL 1.0.2，我们将其输入到生成器中。如果您选择“现代”兼容性模型，这将创建一条 `SSLCipherSuite` 指令，虽然该指令积极实施安全性，但仍适用于大多数浏览器。如果您的软件不支持现代配置，则可以更新软件或改为选择“中间”配置。

```
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
```

选定的密码名称中包含 ECDHE，它是 Elliptic Curve Diffie-Hellman Ephemeral 的缩写。术语 ephemeral 表示向前保密性。顺便说一句，这些密码不支持。 RC4

建议您使用密码的明确列表，而不是依赖于内容不可见的默认值或简短指令。

将生成的指令复制到 `/etc/httpd/conf.d/ssl.conf` 中。

> **Note**
>
> 此处为方便阅读将指令显示为几行，但在复制到 `/etc/httpd/conf.d/ssl.conf` 时，该指令必须位于一行中，并且密码名称之间只有一个冒号（无空格）。

3. 最后，通过删除以下行开头的“#”来取消对该行的注释。

```
#SSLHonorCipherOrder on
```

该指令强制服务器优先使用排名较高的密码，包括（在该示例中）支持向前保密性的密码。启用此指令后，服务器会在回滚到允许的安全性较低的密码之前尝试建立高度安全的连接。

在完成这两个过程后，将更改保存到 `/etc/httpd/conf.d/ssl.conf` 并重新启动 Apache。

如果您在 Qualys SSL Labs 上再次测试该域名，您应该会看到 RC4 漏洞和其他警告已消失，摘要如下所示。

总评	A
证书	100%
协议支持	100%
密钥交换	90%
密码强度	90%

在每次更新 OpenSSL 时，将引入新的密码并删除对旧密码的支持。保留您的 EC2 AL2023 实例 up-to-date，留意来自 OpenSSL 的安全公告，并警惕技术媒体上有关新安全漏洞的报告。

故障排除

- 除非我输入密码，否则我的 Apache Web 服务器不会启动

  如果您安装了受密码保护的加密的私有服务器密钥，这是预期行为。

您可以从密钥中删除加密和密码要求。假设在默认目录中具有一个称为 `custom.key` 的加密的私有 RSA 密钥，并且此密钥上的密码是 **abcde12345**，则对 EC2 实例运行以下命令可生成此密钥的未加密版本。

```
[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 现在启动时应该不会提示您提供密码。

- 我在运行 sudo dnf install -y mod_ssl 时收到了错误。

  在为 SSL 安装所需的程序包时，您可能会看到与以下内容类似的错误。

  ```
  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
  ```

  这通常意味着您的 EC2 实例未运行 AL2023。本教程仅支持从官方 AL2023 AMI 新创建的实例。

教程：在上发布 WordPress 博客 AL2023

以下过程将帮助您在 AL2023 实例上安装、配置和保护 WordPress 博客。AL2023 本教程很好地介绍了如何使用 Amazon EC2，因为您可以完全控制托管 WordPress 博客的 Web 服务器，这在传统托管服务中并不常见。

您负责更新软件包并为您的服务器维护安全补丁。对于不需要与 Web 服务器配置直接交互的自动化程度更高的 WordPress 安装，该 CloudFormation 服务提供了一个可以帮助您快速入门的 WordPress模板。有关更多信息，请参阅AWS CloudFormation 用户指南中的入门。如果您需要具有分离数据库的高可用性解决方案，请参阅《开发人员指南》AWS Elastic Beanstalk 中的部署高可用性 WordPress 网站。

> **Important**
>
> 这些程序旨在与一起使用 AL2023。有关其他发布版本的信息，请参阅特定于该版本的文档。本教程中的很多步骤对 Ubuntu 实例并不适用。有关在 Ubuntu 实例 WordPress 上安装的帮助，请参阅 Ubuntu 文档WordPress中的。你也可以使用CodeDeploy在亚马逊 Linux、macOS 或 Unix 系统上完成此任务。

主题

先决条件

我们强烈建议您将弹性 IP 地址 (EIP) 关联到用于托管 WordPress 博客的实例。这将防止您的实例的公有 DNS 地址更改和中断您的安装。如果您有一个域名且打算将其用于您的博客，则可更新该域名的 DNS 记录，使其指向您的 EIP 地址 (如需帮助，请联系您的域名注册商)。您可以免费将一个 EIP 地址与正在运行的实例相关联。有关更多信息，请参阅《Amazon EC2 用户指南》中的弹性 IP 地址。本 教程：在上安装 LAMP 服务器 AL2023 教程还介绍了配置安全组以允许 HTTP 和 HTTPS 流量的步骤，以及用于确保为 Web 服务器正确设置文件权限的几个步骤。有关向安全组添加规则的信息，请参阅向安全组添加规则。

如果您的博客还没有域名，则可使用 Route 53 注册一个域名并将您的实例的 EIP 地址与您的域名相关联。有关更多信息，请参阅 Amazon Route 53 开发人员指南中的使用 Amazon Route 53 注册域名。

安装 WordPress

连接到您的实例，然后下载 WordPress 安装包。有关连接到实例的更多信息，请参阅连接到 AL2023 实例。

1. 使用以下命令下载并安装这些软件包。

```
dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json
 php php-devel -y
```

2. 您可能会发现显示一条与输出中措辞类似的警告（版本可能随着时间变化）：

```
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
```

   作为最佳实践，我们建议 up-to-date尽可能保留操作系统，但您可能需要遍历每个版本，以确保您的环境中没有冲突。如果步骤 1 中提及的软件包安装失败，则可能需要更新到所列出的某个较新版本，然后重试。

3. 使用wget命令下载最新的 WordPress 安装包。以下命令始终会下载最新版本。

```
[ec2-user ~]$ wget https://wordpress.org/latest.tar.gz
```

4. 解压并解档安装包。安装文件夹解压到名为 `wordpress` 的文件夹。

```
[ec2-user ~]$ tar -xzf latest.tar.gz
```

为您的 WordPress 安装创建数据库用户和数据库

您的 WordPress 安装需要将博客文章和用户评论等信息存储在数据库中。此过程帮助您创建自己的博客数据库，并创建一个有权读取该数据库的信息并将信息保存到该数据库的用户。

1. 启动数据库和 Web 服务器。

```
[ec2-user ~]$ sudo systemctl start mariadb httpd
```

2. 以 `root` 用户身份登录数据库服务器。在系统提示时输入您的数据库 `root` 密码，它可能与您的 `root` 系统密码不同；如果您尚未给您的数据库服务器加密，它甚至可能是空的。

如果您尚未给您的数据库服务器加密，则必须执行这项操作。有关更多信息，请参阅 步骤 3：确保数据库服务器的安全 (AL2023)。

```
[ec2-user ~]$ mysql -u root -p
```

3. 为您的 MySQL 数据库创建用户和密码。您的 WordPress安装使用这些值与您的 MySQL 数据库进行通信。输入以下命令，以替换唯一的用户名和密码。

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

确保为您的用户创建强密码。请勿在您的密码中使用单引号字符 (')，因为这将中断前面的命令。请勿重复使用现有密码，并确保将密码保存在安全的位置。

4. 创建数据库。为数据库提供一个有意义的描述性名称，例如 `wordpress-db`。

> **Note**
>
> 以下命令中数据库名称两边的标点符号称为反引号。在标准键盘上，反引号 (`) 键通常位于 Tab 键的上方。并不总是需要反引号，但是它们允许您在数据库名称中使用其他的非法字符，例如连字符。

```
CREATE DATABASE `wordpress-db`;
```

5. 向之前创建的 WordPress 用户授予数据库的完全权限。

```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

6. 刷新数据库权限以接受您的所有更改。

```
FLUSH PRIVILEGES;
```

7. 退出 `mysql` 客户端。

```
exit
```

创建和编辑 wp-config.php 文件

WordPress 安装文件夹包含一个名为的示例配置文件wp-config-sample.php。在本步骤中，您将复制此文件并进行编辑以适合您的具体配置。

1. 将 wp-config-sample.php 文件复制为一个名为 wp-config.php 的文件。这样做会创建新的配置文件并将原先的示例配置文件原样保留作为备份。

```
[ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
```

2. 用您喜欢的文本编辑器（例如 nano 或 vim）编辑 wp-config.php 文件并输入适用于您的安装的值。如果没有常用的文本编辑器，nano 比较适合初学者使用。

```
[ec2-user ~]$ nano wordpress/wp-config.php
```

   a. 查找定义 DB_NAME 的行并将 database_name_here 更改为您在 Step 4 的 为您的 WordPress 安装创建数据库用户和数据库 中创建的数据库名称。

```
define('DB_NAME', 'wordpress-db');
```

   b. 查找定义 DB_USER 的行并将 username_here 更改为您在 Step 3 的 为您的 WordPress 安装创建数据库用户和数据库 中创建的数据库用户。

```
define('DB_USER', 'wordpress-user');
```

   c. 查找定义 DB_PASSWORD 的行并将 password_here 更改为您在 Step 3 的 为您的 WordPress 安装创建数据库用户和数据库 中创建的强密码。

```
define('DB_PASSWORD', 'your_strong_password');
```

   d. 查找名为 Authentication Unique Keys and Salts 的一节。这些KEY和SALT值为 WordPress 用户存储在本地计算机上的浏览器 Cookie 提供了一层加密。总而言之，添加长的随机值将使您的站点更安全。访问 https://api.wordpress。 org/secret-key/1.1/salt/以随机生成一组密钥值，您可以将其复制并粘贴到wp-config.php文件中。要粘贴文本到 PuTTY 终端，请将光标放在您要粘贴文本的地方，并在 PuTTY 终端内部右键单击鼠标。

      有关安全密钥的更多信息，请访问 https://wordpress。 org/support/article/editing-wp-config-php/#security-keys。

> **Note**
>
> 以下值仅用作示例；请勿使用以下值进行安装。

```
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)XjlkwS1y5NJ76E6EJ.AV0pCKZZB,*~*r ?
6OP$eJT@;+(ndLg');
define('LOGGED_IN_KEY',    'ju}qwre3V*+8f_zOWf?{LlGsQ]Ye@2Jh^,8x>)Y |;(^[Iw]Pi
+LG#A4R?7N`YB3');
define('NONCE_KEY',        'P(g62HeZxEes|LnI^i=H,[XwK9I&[2s|:?0N}VJM%?;v2v]v+;
+^9eXUahg@::Cj');
define('AUTH_SALT',        'C$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-
Es7Q1O-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. 保存文件并退出文本编辑器。

将 WordPress 文件安装在 Apache 文档根目录下

- 现在，您已经解压缩了安装文件夹，创建了 MySQL 数据库和用户，并自定义了 WordPress 配置文件，接下来就可以将安装文件复制到 Web 服务器文档根目录了，这样就可以运行完成安装的安装脚本了。这些文件的位置取决于您是希望 WordPress 博客在 Web 服务器的实际根目录（例如*my.public.dns.amazonaws.com*）中可用，还是在根目录下的子目录或文件夹（例如*my.public.dns.amazonaws.com/blog*）中可用。
  - 如果 WordPress 要在文档根目录下运行，请按如下方式复制 wordpress 安装目录的内容（但不是目录本身）：

    ```
    [ec2-user ~]$ cp -r wordpress/* /var/www/html/
    ```

  - 如果 WordPress 要在文档根目录下的备用目录中运行，请先创建该目录，然后将文件复制到该目录。在此示例中， WordPress将从以下目录运行blog：

```
[ec2-user ~]$ mkdir /var/www/html/blog
[ec2-user ~]$ cp -r wordpress/* /var/www/html/blog/
```

> **Important**
>
> 出于安全原因，如果您不打算立即进入到下一个过程，请立即停止 Apache Web 服务器 (httpd)。将安装移至 Apache 文档根目录下后， WordPress 安装脚本将不受保护，如果 Apache Web 服务器正在运行，攻击者可能会访问您的博客。要终止 Apache Web 服务器，请输入命令 sudo service httpd stop。如果您即将继续到下一个步骤，则不需要终止 Apache Web 服务器。

允许 WordPress 使用永久链接

WordPress 永久链接需要使用 Apache .htaccess 文件才能正常工作，但是 Amazon Linux 上默认不启用此功能。使用此过程可允许 Apache 文档根目录中的所有覆盖。

1. 使用您常用的文本编辑器（如 vim 或 nano）打开 httpd.conf 文件。如果没有常用的文本编辑器，nano 比较适合初学者使用。

   ```
   [ec2-user ~]$ sudo vim /etc/httpd/conf/httpd.conf
   ```

2. 找到以 <Directory "/var/www/html"> 开头的部分。

   ```
   <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. 在以上部分中将 AllowOverride None 行改为读取 AllowOverride *All*。

> **Note**
>
> 此文件中有多个 AllowOverride 行；请确保更改 <Directory "/var/www/html"> 部分中的行。

```
AllowOverride All
```

4. 保存文件并退出文本编辑器。

要在上安装 PHP 图形绘图库 AL2023

PHP 的 GD 库允许您修改图像。如果您需要裁剪博客的标题图像，请安装此库。您安装 phpMyAdmin 的版本可能需要此库的特定最低版本（例如 8.1 版）。

使用以下命令在上安装 PHP 图形绘图库 AL2023。例如，如果您在安装 LAMP 堆栈的过程中从源安装了 php8.1，则此命令将安装 8.1 版的 PHP 图形绘图库。

```
[ec2-user ~]$ sudo dnf install php-gd
```

要验证安装的版本，请使用以下命令：

```
[ec2-user ~]$ sudo dnf list installed | grep php-gd
```

下面是示例输出：

```
php-gd.x86_64                    8.1.30-1.amzn2           @amazonlinux
```

在 Amazon Linux AMI 上安装 PHP 图形绘图库

PHP 的 GD 库允许您修改图像。如果您需要裁剪博客的标题图像，请安装此库。您安装 phpMyAdmin 的版本可能需要此库的特定最低版本（例如 8.1 版）。

要验证哪些版本可用，请使用以下命令：

```
[ec2-user ~]$ dnf list | grep php
```

以下是 PHP 图形绘图库（8.1 版）的输出中的示例行：

```
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
```

使用以下命令在 Amazon Linux AMI 上安装特定版本的 PHP 图形绘图库（例如 php 8.1 版）：

```
[ec2-user ~]$ sudo dnf install -y php8.1-gd
```

修复 Apache Web 服务器的文件权限

中的某些可用功能 WordPress 需要对 Apache 文档根目录具有写入权限（例如通过 “管理” 屏幕上传媒体）。如果您尚未进行此操作，请应用以下组成员关系和权限 (在 LAMP Web 服务器教程中有更为详细的描述)。

1. 将 /var/www 及其内容的文件所有权授予 apache 用户。

   ```
   [ec2-user ~]$ sudo chown -R apache /var/www
   ```

2. 将 /var/www 及其内容的组所有权授予 apache 组。

```
[ec2-user ~]$ sudo chgrp -R apache /var/www
```

3. 更改 /var/www 及其子目录的目录权限，以添加组写入权限及设置未来子目录上的组 ID。

```
[ec2-user ~]$ sudo chmod 2775 /var/www
[ec2-user ~]$ find /var/www -type d -exec sudo chmod 2775 {} \;
```

4. 递归地更改 /var/www 及其子目录的文件权限。

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

> **Note**
>
> 如果您还打算用 WordPress 作 FTP 服务器，则需要在此处进行更宽松的群组设置。请查看中的建议步骤和安全设置 WordPress以完成此操作。

5. 重启 Apache Web 服务器，让新组和权限生效。

```
[ec2-user ~]$ sudo systemctl restart httpd
```

要使用运行 WordPress 安装脚本 AL2023

您已准备好进行安装 WordPress。您使用的命令取决于操作系统。此过程中的命令可用于 AL2023。在 AL2023 AMI 中使用此步骤之后的步骤。

1. 使用 systemctl 命令确保 httpd 和数据库服务在每次系统启动时启动。

```
[ec2-user ~]$ sudo systemctl enable httpd && sudo systemctl enable mariadb
```

2. 验证数据库服务器是否正在运行。

```
[ec2-user ~]$ sudo systemctl status mariadb
```

如果数据库服务未运行，请启动。

```
[ec2-user ~]$ sudo systemctl start mariadb
```

3. 验证您的 Apache Web 服务器 (httpd) 正在运行。

```
[ec2-user ~]$ sudo systemctl status httpd
```

如果 httpd 服务未运行，请启动。

```
[ec2-user ~]$ sudo systemctl start httpd
```

4. 在 Web 浏览器中，键入 WordPress 博客的 URL（要么是您的实例的公有 DNS 地址，要么是该地址后跟blog文件夹）。您应该会看到 WordPress 安装脚本。提供 WordPress 安装所需的信息。选择安装 WordPress以完成安装。有关更多信息，请参阅 WordPress 网站上的步骤 5：运行安装脚本。

使用 AL2023 AMI 运行 WordPress 安装脚本

1. 使用 chkconfig 命令确保 httpd 和数据库服务在每次系统启动时启动。

```
[ec2-user ~]$ sudo chkconfig httpd on && sudo chkconfig mariadb on
```

2. 验证数据库服务器是否正在运行。

```
[ec2-user ~]$ sudo service mariadb status
```

如果数据库服务未运行，请启动。

```
[ec2-user ~]$ sudo service mariadb start
```

3. 验证您的 Apache Web 服务器 (httpd) 正在运行。

```
[ec2-user ~]$ sudo service httpd status
```

如果 httpd 服务未运行，请启动。

```
[ec2-user ~]$ sudo service httpd start
```

4. 在 Web 浏览器中，键入 WordPress 博客的 URL（要么是您的实例的公有 DNS 地址，要么是该地址后跟blog文件夹）。您应该会看到 WordPress 安装脚本。提供 WordPress 安装所需的信息。选择安装 WordPress以完成安装。有关更多信息，请参阅 WordPress 网站上的步骤 5：运行安装脚本。

后续步骤

测试完 WordPress 博客后，可以考虑更新其配置。

使用自定义域名

如果您有一个与您的 EC2 实例的 EIP 地址关联的域名，则可将您的博客配置为使用该域名而不是 EC2 公有 DNS 地址。有关更多信息，请参阅更改网站上的站点 URL。 WordPress

配置您的博客

您可以将您的博客配置为使用不同的主题和插件，从而向您的读者提供更具个性化的体验。但是，有时安装过程可能事与愿违，从而导致您丢失您的整个博客。强烈建议您在尝试安装任何主题或插件之前，为您的实例创建一个备份 Amazon Machine Image (AMI)，以便在安装过程中出现任何问题时，您还可以还原您的博客。有关更多信息，请参阅《Amazon EC2 用户指南》中的创建您自己的 AMI。

增加容量

如果您的 WordPress 博客越来越受欢迎，并且您需要更多的计算能力或存储空间，请考虑以下步骤：

- 对实例扩展存储空间。有关更多信息，请参阅 Amazon EBS 弹性卷。
- 将您的 MySQL 数据库移动到 Amazon RDS 以利用服务的轻松扩展功能。

提高互联网流量的网络性能

如果希望您的博客吸引世界各地用户的流量，请考虑 AWS Global Accelerator。Global Accelerator 通过改善用户的客户端设备和运行的 WordPress 应用程序之间的互联网流量性能，帮助您实现更低的延迟 AWS。Global Accelerator 使用AWS 全球网络将流量引导到离客户端最近的 AWS 区域中运行良好的应用程序终端节点。

了解更多关于 WordPress

以下链接包含有关的更多信息 WordPress。

- 有关信息 WordPress，请参阅 WordPress Codex 上的 Codex 帮助文档。
- 有关安装故障排除的更多信息，请访问常见安装问题。
- 有关提高 WordPress 博客安全性的信息，请访问强化 WordPress。
- 有关保留 WordPress 博客的信息 up-to-date，请前往更新 WordPress。

帮助！我的公有 DNS 名称发生更改导致我的博客瘫痪

您的 WordPress 安装是使用您的 EC2 实例的公有 DNS 地址自动配置的。如果您停止并重启实例，公有 DNS 地址将发生更改 (除非它与弹性 IP 地址相关联)，并且您的博客将不会再运行，因为您的博客引用了不再存在的地址 (或已分配给另一个 EC2 实例的地址) 上的资源。https://wordpress 中概述了对该问题的更详细描述和几种可能的解决方案。 org/support/article/changing-the-site-url/。

如果 WordPress 安装时发生了这种情况，则可以通过以下步骤恢复博客，该过程使用wp-cli命令行界面 WordPress。

要使用更改您的 WordPress 网站网址 wp-cli

1. 使用 SSH 连接到您的 EC2 实例。
2. 请记下您的实例的旧站点 URL 和新站点 URL。安装时，旧站点 URL 很可能是您的 EC2 实例的公有 DNS 名称 WordPress。新站点 URL 是您的 EC2 实例的当前公有 DNS 名称。如果您不确定旧站点 URL 是什么，则可通过以下命令使用 curl 来查找它。

   ```
   [ec2-user ~]$ curl localhost | grep wp-content
   ```

   您应该会在输出中看到对您的旧公有 DNS 名称的引用，如下所示 (旧站点 URL 用红色表示)：

   ```
   <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. 使用以下命令下载 wp-cli。

   ```
   [ec2-user ~]$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/
   phar/wp-cli.phar
   ```

4. 使用以下命令搜索并替换 WordPress 安装中的旧站点 URL。用新旧站点 URL 替换您的 EC2 实例和 WordPress 安装路径（通常为 o /var/www/html r/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. 在 Web 浏览器中，输入 WordPress 博客的新网站 URL，以验证该网站是否恢复正常运行。如果不是，请参阅更改站点 URL 和常见安装问题以获取更多信息。

教程：023 年 Redis 6 到 Valkey 过渡 AL2

以下文档描述了 023 年从 Redis 6 过渡到 Valkey 的关键方面 AL2。

Redis 6 的支持时间线

Redis 6 将于 2027 年 1 月 31 日进入生命尽头 (EOL)。在此日期之后，Redis 6 将不再从 Redis 项目获得更新或安全补丁。我们强烈建议用户在 2027 年 1 月之前迁移到 Valkey，以确保持续的支持和安全更新。

有关 Redis 版本支持时间表的更多信息，请参阅 Redis End-Of-Life 计划文档。

Valkey 简介

Valkey 是 Redis 7 的一个开源分支，由 Linux 基金会维护。它与 Redis 开源软件（OSS）2.x 到 7.2.x 版本完全兼容。Valkey 保持了熟悉的 Redis API 和功能，同时提供了若干增强：

- 通过多线程提升性能。
- 提高内存效率，尤其是在集群模式下。
- 双通道复制以实现更好的数据一致性。

迁移计划和时间线

强烈建议用户在 2027 年 1 月 31 日 Redis 6 进入生命周期 (EOL) 之前从 Redis 6 迁移到 Valkey。此迁移需要手动干预，不是自动的。

Amazon Linux 建议进行此迁移，以确保您依赖 Redis 的应用程序具有持续的功能、支持和安全更新。

迁移选项和步骤

我们根据您的部署需求和运营需求，提出了三条迁移至 Valkey 的路径。

选项 1：新实例安装

适用于新部署或不需要数据迁移的情况：

1. 安装 Valkey：

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

2. 启动 Valkey：

```
[ec2-user ~]$ sudo systemctl start valkey
```

3. （可选）设置开机自启 Valkey：

```
[ec2-user ~]$ sudo systemctl enable valkey
```

4. 验证安装：

```
[ec2-user ~]$ valkey-cli info server
[ec2-user ~]$ valkey-cli ping
```

选项 2：原地替换

适用于不需要数据持久化的现有实例：

1. 停止 Redis 6：

```
[ec2-user ~]$ sudo systemctl stop redis6
```

2. 安装 Valkey：

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

3. （可选）在 Valkey 中使用 Redis 6 配置：

```
[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. （可选）在 Valkey 中使用 Redis 6 哨兵配置文件：

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

5. 启动 Valkey：

```
[ec2-user ~]$ sudo systemctl start valkey
```

6. （可选）设置开机自启 Valkey：

```
[ec2-user ~]$ sudo systemctl enable valkey
```

7. 验证 Valkey 安装：

```
[ec2-user ~]$ valkey-cli info server
[ec2-user ~]$ valkey-cli ping
```

8. 移除 Redis 6：

```
[ec2-user ~]$ sudo dnf remove redis6
```

选项 3：数据迁移

此选项允许您同时运行 Redis 6 和 Valkey。

1. 安装 Valkey 但不移除 Redis 6：

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

2. （可选）在 Valkey 中使用 Redis 6 配置：

```
[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. （可选）在 Valkey 中使用 Redis 6 哨兵配置文件：

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

4. 修改 Valkey 配置：

   编辑 `/etc/valkey/valkey.conf` 并将“port”指令设置为不同的值（例如 6380），以避免与 Redis 6 冲突。

5. 启动 Valkey：

```
[ec2-user ~]$ sudo systemctl start valkey
```

6. （可选）设置开机自启 Valkey：

```
[ec2-user ~]$ sudo systemctl enable valkey
```

7. 验证 Valkey 安装：

```
[ec2-user ~]$ valkey-cli -p port info server
[ec2-user ~]$ valkey-cli -p port ping
```

> **Note**
>
> *port*替换为配置的端口号。

8. 迁移数据：

   您现在可以使用复制或手动数据传输方法将数据从 Redis 6 迁移到 Valkey。

9. 更新应用程序配置：

   逐步更新您的应用程序以使用 Valkey 端口。

10. 移除 Redis 6：

    一旦所有数据和应用程序都已迁移，您可以停止并移除 Redis 6。

```
[ec2-user ~]$ sudo systemctl stop redis6
[ec2-user ~]$ sudo dnf remove redis6
```

> **Note**
>
> 强烈建议在生产系统中实施更改之前，在测试环境中验证迁移过程。

相关主题

有关 Valkey 的更多信息：

- Valkey：https://valkey.io/
- Valkey 迁移：https://valkey。 io/topics/migration/

教程：在上安装 GNOME 桌面环境 AL2023

自 2023.7 版或更高版本起，GNOME 桌面环境可 AL2023 作为可选的图形用户界面提供。

以下过程可帮助您在 AL2023实例上安装 GNOME 桌面环境。您可以使用此图形界面，通过熟悉的桌面环境与您的 Linux 系统交互，而不仅仅是命令行界面。

内容

前提条件

- 桌面环境至少需要 2.4 GB 的内存。因此，建议使用 `t2.medium` 或更高类型的实例以确保足够的性能。内存不足的实例类型示例包括 `t2.nano`、`t2.micro` 和 `t2.small`。此限制也适用于此大小的 `t3` 和 `t4` 实例，以及任何其他不满足内存要求的实例类型。
- 本教程假设您已经使用 AL2023 运行版本 2023.7 或更高版本启动了实例。更多信息，请参阅 Amazon EC2 上的 AL2023 和 正在更新 AL2023 页面。

安装

- 安装 GNOME 桌面环境及相关程序包。

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

> **Note**
>
> 要访问图形桌面环境，您需要安装并配置其他软件，例如 Amazon DCV 或 VNC。这些工具允许您通过网络连接并与图形用户界面进行交互。

相关主题

有关图形桌面环境的更多信息，请参阅以下文档：

- 《Amazon DCV 管理员指南》中的什么是 Amazon DCV？
- 教程：在上配置 TigerVNC 服务器 AL2023

教程：在上配置 TigerVNC 服务器 AL2023

以下过程可帮助您在 AL2023 实例上设置 VNC 服务器。VNC 允许您通过安全的网络连接远程访问图形桌面环境并与之交互。

内容

前提条件

- 本教程假设您已经在 AL2023 实例上安装了 GNOME 桌面环境。有关更多信息，请参阅教程：在上安装 GNOME 桌面环境 AL2023页面。
- 本教程使用 SSH 端口转发来访问 VNC 服务器。有关设置密钥对的更多信息，请参阅《Amazon EC2 用户指南》中的使用 SSH 连接到您的 Linux 实例。
- 以下步骤不指导您安装 VNC 客户端的过程。您必须在本地计算机上安装 VNC 客户端，才能连接至桌面环境并与之交互。

步骤 1：安装

1. 连接到您的实例。有关更多信息，请参阅 连接到 AL2023 实例。
2. 安装适用的 TigerVNC 服务器软件包。 AL2023

-y 选项将在不要求确认的情况下安装程序包。如果您希望在安装前检查程序包，可以省略此选项。

```
[ec2-user ~]$ sudo dnf install -y tigervnc-server
```

步骤 2：配置

1. 确保用户已配置 VNC 密码。

```
[ec2-user ~]$ vncpasswd
```

2. 为用户分配一个显示编号。

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver.users
```

添加以下配置：

```
:1=ec2-user
```

> **Note**
>
> 您可以为用户分配任意显示编号。在此示例中，我们使用显示编号 :1。

3. 编辑 VNC 服务器配置文件。

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver-config-defaults
```

添加以下配置：

```
session=gnome
securitytypes=vncauth,tlsvnc
geometry=1920x1080
localhost
alwaysshared
```

> **Note**
>
> 您可以使用 geometry 参数更改显示分辨率。在此示例中，我们使用 1920x1080。

4. 启动 VNC 服务器。每次重启实例后都需要重复此过程。如果您希望自动化启动此服务的过程，请参阅下面的可选部分。

```
[ec2-user ~]$ sudo systemctl start vncserver@:1
```

> **Important**
>
> 启动 vncserver 服务时， @ 后面的部分必须与在 /etc/tigervnc/vncserver.users 文件中为用户设置的显示编号匹配。

执行此步骤后，您可以从本地计算机创建 SSH 隧道，并使用 VNC 客户端进行连接。

步骤 3：使用 VNC 客户端连接

VNC 服务器公开一个 TCP 套接字供客户端连接。虽然您可以直接通过安全组公开 VNC 端口，但本教程演示了使用 SSH 隧道作为一种更安全的方法，通过加密本地计算机与 EC2 实例之间的连接来实现。通过隧道连接后，您将使用上一步中配置的密码向 VNC 服务器进行身份验证。有关安全组的更多信息，请参阅《Amazon EC2 用户指南》中的更改 Amazon EC2 实例的安全组。

1. 从本地计算机创建 SSH 隧道。

```
$ ssh -i <keypair> -L 5901:localhost:5901 ec2-user@<address>
```

> **Note**
>
> 将 <keypair> 替换为您的 SSH 密钥路径，并将 <address> 替换为您实例的公共 IP 或 DNS 名称。端口号根据用于启动 vncserver 的显示编号而变化。例如，显示编号 :1 使用端口 5901，显示编号 :2 使用端口 5902，依此类推。

2. 使用您的 VNC 客户端连接到 localhost:5901 或 127.0.0.1:5901，并使用先前设置的 VNC 密码。

> **Important**
>
> 使用 VNC 时请保持 SSH 隧道处于打开状态。如果 SSH 隧道未打开，您将无法使用 VNC 客户端查看桌面环境并与之交互。

（可选）启动时启动服务

如果您计划定期使用 VNC，可能需要配置 VNC 服务器在实例启动时自动启动。这消除了每次重启实例后手动启动 VNC 服务器的需要。此配置确保您的图形桌面环境在实例完成启动过程后立即可用并可访问。

- 配置服务在启动时启动。

```
[ec2-user ~]$ sudo systemctl enable vncserver@:1
```

> **Important**
>
> 启用 vncserver 服务时， @ 后面的部分必须与在 /etc/tigervnc/vncserver.users 文件中为用户设置的显示编号匹配。此外，您可以在 enable 之后传递 --now 参数以立即启动服务。

执行此步骤后，您将不再需要在每次重启实例时启动 vncserver。

（可选）禁用空闲锁屏

- 将空闲延迟设置为零，以便在用户长时间不活动时禁用锁屏。

```
[ec2-user ~]$ gsettings set org.gnome.desktop.session idle-delay 0
```

相关主题

有关图形桌面环境的更多信息，请参阅以下文档：

- 教程：在上安装 GNOME 桌面环境 AL2023

- 《Amazon DCV 管理员指南》中的什么是 Amazon DCV？

在内核上使用多代 LRU (MGLRU) AL2023

多代 LRU 是 Linux 内核中的一种现代页面回收算法，旨在提高内存压力下的内存管理性能。它取代了传统的 LRU（最近最少使用）机制，该机制用于在系统内存不足时确定要回收哪些内存页。

传统的 LRU 机制使用双列表模型（活跃和非活跃）来跟踪页面使用情况，这在具有大型工作集的现代工作负载中可能效率低下。MGLRU 用多“代”页面取代了这种机制，允许内核基于更细粒度的老化信息做出更智能的决策。

MGLRU 的好处包括：

- 更好的回收决策：更准确地识别冷（未使用）页面。
- 更低的延迟和更高的吞吐量：特别是对于具有大地址空间或许多并发进程的工作负载。
- 改进的缓存保留：最近使用的页面不太可能被过早换出。
- 可扩展且锁定效率高的设计：在有许多机器上性能更好。 CPUs

配置和调优

内核配置已`CONFIG_LRU_GEN`在内 AL2023 核上启用。这会编译进 MGLRU，但默认不启用它。

可以使用 `/sys/kernel/mm/lru_gen/enabled` 文件启用和调整 MGLRU。该值是一个位掩码。建议启用所有组件，除非其中某些组件存在不良副作用。

Bit	组件
`0`	多代 LRU 的主开关。
1	当 MMU 设置时（例如在 x86 架构上），以大批量方式清除叶子页表项中的访问位。此行为理论上可能加剧锁争用 (`mmap_lock`)。如果禁用此功能，对于连续映射热页的工作负载，多代 LRU 将遭受轻微的性能下降，因为这些页面的访问位本可以通过较少的大批量操作清除。
2	当 MMU 设置时（例如在 x86 架构上），同时清除非叶子页表项中的访问位。此行为未在 Intel

Bit	组件
	和 AMD 以外的 x86 变体上得到验证。如果禁用此功能，多代 LRU 将遭受可忽略不计的性能下降。
[yYnN]	Enable/disable 以上所有组件。

如何启用 MGLRU 的示例：

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

此命令启用所有组件：

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

在 Amazon EC2 之外使用 Amazon Linux 2023

Amazon Linux 2023 容器映像可以在兼容的容器运行时环境中运行。有关如何在容器内使用 Amazon Linux 2023 的更多信息，请参阅AL2023 在容器里。

Amazon Linux 2023 (AL2023) 除了可以直接在 Amazon EC2 上运行外，还可以作为虚拟客户机运行。目前提供 KVM (qcow2)、VMware (OVA) 和 Hyper-V (vhdx) 映像。

> **Note**
>
> Amazon Linux 2023 的映像配置不同于 Amazon Linux 2。
> 如果您执行了在本地以虚拟机形式运行 Amazon Linux 2，则需要调整配置来兼容 AL2023。

下载适用于 KVM、VMware 和 Hyper-V 的 Amazon Linux 2023 映像

适用于 KVM、VMware 和 Hyper-V 的 Amazon Linux 2023 磁盘映像可从 cdn.amazonlinux.com 下载。

在非 Amazon EC2 虚拟化环境中使用 Amazon Linux 2023 的支持的配置

这部分介绍在非 Amazon EC2 虚拟化环境（例如在 KVM、VMware 或 Hyper-V 上）中运行 Amazon Linux 2023 的要求。

基本的 AL2023 系统要求适用于所有非 Amazon EC2 虚拟化环境。以下主题详细列出对于每个虚拟机监控器环境所支持的设备型号列表。

KVM、VMware 和 Hyper-V 提供了许多配置选项，需要根据您的安全性、性能和可靠性需求仔细配置它们。有关更多信息，请查看虚拟机监控器提供的文档。

主题

- 在 KVM AL2023 上运行的要求
- 在 VMware 上运行 AL2023 的要求
- 在 Hyper-V 上运行 Amazon Linux 2023 的要求

在 KVM AL2023 上运行的要求

本节介绍在 KVM AL2023 上运行的要求。的 KVM 映像适用于 AL2023 两种x86-64架构aarch64。这些要求是 KVM 映像基础 AL2023 系统要求 之外的附加要求。

主题

在 KVM AL2023 上运行的 KVM 主机要求

KVM 映像目前已在运行 Ubuntu 22.04.3 LTS 的主机上通过认证，该主机使用此 Ubuntu 版本提供的 qemu 版本 6.2+dfsg-2ubuntu6.15，并为 x86-64 使用 q35 机器类型，为 aarch64 使用 virt 机器类型。

KVM AL2023 上的设备支持

经测试可与 AL2023 KVM 映像配合使用的**qemu**设备型号（两者**aarch64**兼有**x86-64**）有：

- virtio-blk（virtio 块设备）
- virtio-scsi（带磁盘设备的 virtio SCSI 控制器）
- virtio-net（virtio 网络设备）
- ahci（用于虚拟 CD-ROM 驱动器）
- usb-storage（通过 xhci）

在 AL2023 KVM 映像认证中启用但未大量使用的其他**qemu**设备型号有：

- 仅 x86-64 上的 VGA (qemu VGA)
- virtio-rng（虚拟随机数生成器）
- 传统 AT 键盘和 PS/2 鼠标设备
- 传统串行设备

KVM AL2023 上的启动模式（UEFI和BIOS）支持

x86-64 映像经过了传统 BIOS 和 UEFI 启动模式的测试。aarch64 映像经过了 UEFI 启动模式的测试。

> **Note**
>
> 默认情况下，当使用 UEFI 启动模式时，一些虚拟机管理器将使用 Microsoft 安全启动密钥配置虚拟机，从而启用安全启动。此配置将无法启动 AL2023。
> 由于 AL2023 启动加载程序未由 Microsoft 签名，因此必须在没有 UEFI 密钥或使用安全启动密钥的情况下配置虚拟机。 AL2023

> **Important**
>
> KVM 映像的安全启动支持尚未经过验证。

在 KVM AL2023 上运行的限制

在 KVM AL2023 上运行存在一些已知的限制。

> **Note**
>
> 实现某些列出的不支持的功能的代码可能存在于中 AL2023并且可以正常运行。提供不支持功能列表是为了让您能就当前可依赖的功能以及 Amazon Linux 团队将在未来更新中认证可用的功能做出明智决策。

AL2023 在 KVM 上运行的已知限制

- KVM 客户机代理目前未打包或不支持。
- 不支持热插拔 CPU、内存或任何其他设备类型。
- 不支持虚拟机休眠。
- 不支持虚拟机迁移。
- 不支持任何设备的透传，例如通过 PCI 透传或 USB 透传都不被支持。

在 VMware 上运行 AL2023 的要求

这部分介绍在 VMware 上运行 AL2023 的要求。AL2023 的 VMware 映像仅适用于 x86-64 架构。不支持或未提供适用于 aarch64 的 VMware 映像。这些要求是 VMware 映像基础 AL2023 系统要求 之外的附加要求。

主题

在 VMware 上运行 AL2023 的 VMware 主机要求

AL2023 VMware OVA 映像目前已在以下环境中通过认证：

- 在使用 Intel(R) Xeon(R) Platinum 8124M 处理器的主机上运行的 VMware Workstation 17.5.0
- 使用 Intel(R) Xeon(R) Platinum 8275CL 处理器的 VMware vSphere 8.0

AL2023 VMware OVA 映像指定的机器硬件版本为 13。

支持 VMware 机器硬件版本 13 的环境包括：

- ESXi 6.5 或更高版本
- VMware Workstation 14 或更高版本

AL2023 在 VMware 上的设备支持

以下 VMware 设备模型已测试可用于 AL2023 VMware OVA 映像（仅限 **x86-64**）：

- `vmw_pvscsi`（VMware 准虚拟化 SCSI 控制器）
- `vmxnet3`（VMware 准虚拟化网络设备）
- `ata_piix`（传统 IDE 仅适用于虚拟 CD-ROM 驱动器）

在 AL2023 VMware 映像认证中启用但未经过充分测试的其他 VMware 设备模型：

- `vmw_vmci` 及相关 `vsock` 接口（用于 VMware 客户机代理的虚拟套接字传输）
- `vmw_balloon` 内存气球设备
- VMware SVGA 控制器
- 传统 AT 键盘和 PS/2 鼠标设备

VMware 客户机代理程序包 (`open-vm-tools`) 在 AL2023 VMware OVA 映像中默认可用且已安装。

AL2023 在 VMware 上的启动模式支持（UEFI 和 BIOS）

自 2023.3.20231211 发布版起，AL2023 VMware OVA 映像已在传统 BIOS 和 UEFI 启动模式下通过验证。OVA 默认配置仍为传统 BIOS，但可由用户更改。

> **Important**
>
> 安全启动支持需要 UEFI，该功能尚未在运行于 VMware 上的 AL2023 上经过验证。

在 VMware 上运行 AL2023 的限制

在 VMware 上运行 AL2023 存在一些已知限制。

> **Note**
>
> AL2023 中可能存在实现所列的一些不被支持的功能的代码，并且可以正常运行。不被支持的功能列表之所以存在，是为了让客户能够就什么依赖当前工作，以及 Amazon Linux 团队将把什么视为未来更新的一部分工作做出明智决定。

在 VMware 上运行 AL2023 的已知限制

- UEFI 安全启动目前在 VMware 上的 AL2023 中未经验证。
- 不支持热插拔 CPU、内存或任何其他设备类型。
- 不支持虚拟机休眠。
- 不支持虚拟机迁移。
- 不支持任何设备的透传，例如通过 PCI 透传或 USB 透传都不被支持。

在 Hyper-V 上运行 Amazon Linux 2023 的要求

这部分介绍在 Hyper-V 上运行 Amazon Linux 2023 的要求。AL2023 的 Hyper-V 映像仅适用于 x86-64 架构。目前不提供或不支持 aarch64 的 Hyper-V 映像。

这部分介绍 Hyper-V 映像在基础 AL2023 系统要求 之上的附加要求。

主题

在 Hyper-V 上运行 Amazon Linux 2023 的 Hyper-V 主机要求

Amazon Linux 2023 在 Hyper-V 上的主要认证是在 EC2 c5.metal 实例上运行的 Windows Server 2022。

Amazon Linux 2023 在 Hyper-V 上的设备支持

Amazon Linux 2023 在具有以下虚拟化硬件集的第 1 代和第 2 代 Hyper-V 虚拟机上进行了测试：

- 第 1 代（传统 BIOS 启动）虚拟机
- 第 2 代（UEFI 启动 - 无安全启动）虚拟机
- 以下设备模型已测试可用于 AL2023 Hyper-V 映像：
  - 用于第 2 代虚拟机根磁盘和模拟 CD-ROM 驱动器的 Hyper-V 虚拟存储 hv_storvsc
  - 用于第 1 代虚拟机虚拟 CD-ROM 驱动器的模拟 PIIX IDE ata_piix
  - Hyper-V 虚拟以太网 hv_netvsc
- 以下设备模型已启用但测试较少：
  - 第 1 代虚拟机上的传统 VGA 文本模式
  - 第 2 代虚拟机上基于 UEFI 固件的帧缓冲器 simpledrmfb
  - Hyper-V Balloon hv_balloon
  - Hyper-V HID/鼠标 hid_hyperv
- 以下设备模式目前未在 AL2023 中启用：
  - Hyper-V PCI 透传
  - Hyper-V DRM 图形

> **Important**
>
> 对于第 2 代虚拟机，不支持安全启动，并且必须在启动虚拟机之前禁用它，以确保 Amazon Linux 2023 成功启动。Hyper-V 目前仅支持使用微软自有密钥签名的软件组件的安全启动，而 Amazon Linux 引导程序由亚马逊私钥签名。Hyper-V 目前不支持导入第三方密钥。

在 Hyper-V 上运行 Amazon Linux 2023 的限制

以下是在 Hyper-V 上运行 Amazon Linux 2023 的一些已知限制：

> **Note**
>
> AL2023 中可能存在实现所列的一些不被支持的功能的代码，并且可以正常运行。不被支持的功能列表之所以存在，是为了让客户能够就什么依赖当前工作，以及 Amazon Linux 团队将把什么视为未来更新的一部分工作做出明智决定。

在 Hyper-V 上运行 AL2023 的已知限制

- UEFI 安全启动模式目前在 Hyper-V 上的 AL2023 上不受支持也无法使用
- 不支持热插拔 CPU、内存或任何其他设备类型。
- 不支持虚拟机休眠。
- 不支持虚拟机迁移。
- 不支持任何设备的透传，例如通过 PCI 透传或 USB 透传都不被支持。

在 Amazon EC2 之外使用时 Amazon Linux 2023 的设置和 `cloud-init` 配置

这部分介绍不在 Amazon EC2 上直接运行时（例如在 KVM、VMware 或 Hyper-V 上）如何设置和配置 Amazon Linux 2023 虚拟机。

默认情况下，Amazon Linux 2023 虚拟机映像没有预置任何用户密码或 ssh 密钥，并且将在第一个被发现的网络接口上通过 DHCP 获得其网络配置。这意味着在默认情况下，没有其他配置，无法连接到生成的虚拟机。

因此，需要向虚拟机提供某种形式的配置。对于 Amazon Linux 来说，完成这项工作的标准机制是通过 `cloud-init` 数据源。

Amazon Linux 2023 已通过以下数据源的认证：

NoCloud

这是配置本地映像的传统方法，即通过包含一个种子 ISO9660 映像和 `cloud-init` 配置文件的虚拟 CD-ROM 来配置。

VMware

Amazon Linux 2023 还支持通过 VMware 特定数据源配置在 vSphere 上运行的 VMware 映像，即通过 `guestinfo.userdata` 和 `guestinfo.metadata` 来配置。

> **Note**
>
> 数据源的配置与 Amazon Linux 2 不同。更具体地说，Amazon Linux 2023 使用 `systemd-networkd` 来进行配置，并且要求使用 `cloud-init`“网络配置版本 2”，如 cloud-init 网络配置文档所述。

有关在 Amazon Linux 2023 中打包的 `cloud-init` 版本的 `cloud-init` 配置机制的完整文档，可从上游 `cloud-init` 文档中找到。

KVM 和 VMware 上的 Amazon Linux 2023 的 NoCloud (`seed.iso`) `cloud-init` 配置

这部分介绍如何创建和使用 `seed.iso` 映像来配置在 KVM 或 VMware 上运行的 Amazon Linux 2023。由于 KVM 和 VMware 环境没有 Amazon EC2 实例元数据服务 (IMDS)，因此需要一种配置 Amazon Linux 2023 的替代方法，而提供 `seed.iso` 映像就是其中一种方法。

`seed.iso` 启动映像包括启动和配置新的虚拟机所需的初始配置信息，例如网络配置、主机名和用户数据。

> **Note**
>
> `seed.iso` 映像仅包括启动虚拟机所需的配置信息，它不包括 Amazon Linux 2023 操作系统文件。

要生成 seed.iso 映像，您至少需要两个配置文件，有时则需要三个：

meta-data

此文件通常包含虚拟机的主机名。

user-data

此文件通常配置用户账户、用户密码、ssh 密钥对和/或访问机制。默认情况下，Amazon Linux 2023 KVM 和 VMware 映像会创建一个 ec2-user 用户账户。您可以使用 user-data 配置文件为此默认用户账户设置密码和/或 ssh 密钥。

network-config（可选）

此文件通常为虚拟机提供网络配置，该配置将覆盖默认配置。默认配置是在第一个可用网络接口上使用 DHCP。

创建 seed.iso 磁盘映像

1. 在 Linux 或 MacOS 计算机上，创建一个名为 seedconfig 的新文件夹并导航到该文件夹。

   > **Note**
   >
   > 虽然可以使用 Windows 或其他操作系统完成这些步骤，但您必须找到 mkisofs 的等效工具才能完成创建 seed.iso 映像。

2. 创建 meta-data 配置文件。

   a. 创建名为 meta-data 的新文件。

   b. 使用首选编辑器打开 meta-data 文件并添加以下内容，将 *vm-hostname* 替换为虚拟机的主机名：

   ```
   #cloud-config
   local-hostname: vm-hostname
   ```

   c. 保存并关闭 meta-data 配置文件。

3. 创建 user-data 配置文件。

   a. 创建名为 user-data 的新文件。

   b. 使用首选编辑器打开 user-data 文件并添加以下内容，进行必要的替换：

```
#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. 您可以选择性地向 user-data 配置文件中添加更多用户账户。

   您可以指定其他用户账户、用户访问机制、密码和密钥对。有关支持的指令的更多信息，请参阅上游 cloud-init 文档。

d. 保存并关闭 user-data 配置文件。

4. （可选）创建 network-config 配置文件。

a. 创建名为 network-config 的新文件。

b. 使用首选编辑器打开 network-config 文件并添加以下内容，将各个 IP 地址替换为适合您的设置的地址。

```
#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 网络配置提供了与接口的 MAC 地址匹配的机制，而不是指定接口名称，接口名称可能会根据虚拟机配置而变化。上游 cloud-init 网络配置版本 2 文档中详细描述了网络配置的这个（以及更多）cloud-init 功能。

c. 保存并关闭 network-config 配置文件。

5. 使用在前面步骤中创建的 meta-data、user-data 和可选的 network-config 配置文件创建 seed.iso 磁盘映像。

   根据要在其上创建 seed.iso 磁盘映像的操作系统，执行以下操作之一。

   - 在 Linux 系统上，使用 **mkisofs** 或 **genisoimage** 之类的工具来创建已完成的 seed.iso 文件。导航到 seedconfig 文件夹，并运行以下命令：

     ```
     $ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
     ```

   - 如果您使用 network-config，则将其包含在 **mkisofs** 调用中：

     ```
     $ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
      network-config
     ```

   - 在 macOS 系统上，您可以使用 **hdiutil** 之类的工具来生成已完成的 seed.iso 文件。由于 **hdiutil** 采用路径名而不是文件列表，因此无论是否创建了 network-config 配置文件，都可以使用相同的调用。

     ```
     $ hdiutil makehybrid -o seed.iso -hfs -joliet -iso -default-volume-name cidata
      seedconfig/
     ```

6. 现在，生成的 seed.iso 文件可以通过虚拟 CD-ROM 驱动器附加到您的新 Amazon Linux 2023 虚拟机，以便在首次启动时被 cloud-init 找到并将配置应用到系统。

VMwareon 的 guestinfo **cloud-init** 配置 AL2023 VMware

VMware环境没有 Amazon EC2 实例元数据服务 (IMDS)，因此需要另 AL2023 一种配置方法。这部分介绍如何使用 VMware vSphere 中可用的 seed.iso 虚拟 CD-ROM 驱动器的替代配置机制。

此配置方法使用 VMware extraconfig 机制向 cloud-init 提供配置数据。对于以下每个键，必须提供相应的 ***keyname*.encoding** 属性。

以下键可提供给 VMware extraconfig 机制。

guestinfo.metadata

包含 cloud-init 元数据的 JSON 或 YAML

guestinfo.userdata

包含 cloud-config 格式的 cloud-init 用户数据的 YAML 文档。

guestinfo.vendordata（可选）

YAML 包含 cloud-init 供应商数据

相应的编码属性（guestinfo.metadata.encoding、guestinfo.userdata.encoding 和 guestinfo.vendordata.encoding）可以包含：

base64

属性的内容用 base64 编码。

gzip+base64

属性的内容先用 base64 编码再用 gzip 压缩。

> **Note**
>
> seed.iso 方法支持一个独立的（可选的）network-config 配置文件。VMware guestinfo 在网络配置的提供方式上有所不同。更多信息在以下部分提供。

如果需要一个明确的网络配置，则应以两个 YAML 或 JSON 属性的形式将其嵌入到 metadata 中：

network

包含 JSON 或 YAML 格式的编码网络配置。

network.encoding

包含上述网络配置数据的编码。对于 guestinfo 数据，cloud-init 支持的编码是相同的：base64 和 gzip+base64。

Example使用 VMware vSphere govc CLI 工具通过 guestinfo 传递配置

1. 按照 KVM 和 VMware 上的 Amazon Linux 2023 的 NoCloud (seed.iso) cloud-init 配置 中的说明准备 meta-data、user-data 和可选的 network-config 配置文件。
2. 将配置文件转换为 VMware guestinfo 可用的格式。

```
# '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"
```

比较安装在 Amazon Linux 2023 标准 AMI 与 AL2023 KVM 映像上的程序包

将 AL2023 标准 AMI 上存在的 RPM 与 AL2023 KVM 镜像上的 RPM 进行了比较。

软件包	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

软件包	AMI	KVM
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
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	2.1.1	2.1.1
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28

软件包	AMI	KVM
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
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-onprem		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

软件包	AMI	KVM
cracklib-dicts	2.9.6	2.9.6
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

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
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-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)

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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-repo-cdn		2023.6.20241031
kernel-livepatch-repo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-extra-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

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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-whence	20210208 (noarch)	20210208 (noarch)
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 (x86_64)	2.1 (x86_64)
mpfr	4.1.0	4.1.0

软件包	AMI	KVM
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-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

软件包	AMI	KVM
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-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

软件包	AMI	KVM
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-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

软件包	AMI	KVM
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
perl-Text-ParseWords	3.30	3.30

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
python3-jsonschema	3.2.0	3.2.0

软件包	AMI	KVM
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-libselinux	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
python3-pyserial	3.4	3.4

软件包	AMI	KVM
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-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

软件包	AMI	KVM
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	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

软件包	AMI	KVM
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
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

软件包	AMI	KVM
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
xfsprogs	5.18.0	5.18.0
xxd	9.0.2153	9.0.2153

软件包	AMI	KVM
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-defaults	1.1.2	
zstd	1.5.5	1.5.5

比较安装在 Amazon Linux 2023 标准 AMI 与 AL2023 VMware OVA 映像上的程序包

将 AL2023 标准 AMI 上存在的 RPM 与 AL2023 VMware OVA 镜像上的 RPM 进行了比较。

软件包	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

软件包	AMI	VMware OVA
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
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	2.1.1	2.1.1
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28

软件包	AMI	VMware OVA
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
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-onprem		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

软件包	AMI	VMware OVA
cracklib-dicts	2.9.6	2.9.6
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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-repo-cdn		2023.6.20241031
kernel-livepatch-repo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-extra-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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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-whence	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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
python3-dnf-plugins-core	4.3.0	4.3.0
python3-docutils	0.16	0.16
python3-gpg	1.15.1	1.15.1

软件包	AMI	VMware OVA
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-libselinux	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

软件包	AMI	VMware OVA
python3-policycoreutils	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
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

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
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
systemd-libs	252.23	252.23
systemd-networkd	252.23	252.23
systemd-pam	252.23	252.23
systemd-resolved	252.23	252.23

软件包	AMI	VMware OVA
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
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

软件包	AMI	VMware OVA
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfsprogs	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
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

比较安装在 Amazon Linux 2023 标准 AMI 与 AL2023 Hyper-V 映像上的程序包

AL2023 标准 AMI 上存在的 RPM 与 AL2023 Hyper-V 映像上存在的 RPM 的比较。

软件包	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
amazon-linux-repo-cdn		2023.4.20240319
amazon-linux-repo-s3	2023.4.20240319	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-network		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

软件包	AMI	Hyper-V VHDX
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	2.1.1	2.1.1
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
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

软件包	AMI	Hyper-V VHDX
cloud-init	22.2.2	22.2.2
cloud-init-cfg-ec2	22.2.2	
cloud-init-cfg-onprem		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-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

软件包	AMI	Hyper-V VHDX
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
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5

软件包	AMI	Hyper-V VHDX
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
file	5.39	5.39
file-libs	5.39	5.39

软件包	AMI	Hyper-V VHDX
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
gnupg2-minimal	2.3.7	2.3.7
gnutls	3.8.0	3.8.0

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
hwdata	0.353	0.353
hyperv-daemons		0
hyperv-daemons-license		0
hypervfcopyd		0
hypervkvpd		0
hyperv-tools		0
hypervvssd		0
info	6.7	6.7
inih	49	49
initscripts	10.09	10.09
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

软件包	AMI	Hyper-V VHDX
kernel-livepatch-repo-cdn		2023.4.20240319
kernel-livepatch-repo-s3	2023.4.20240319	
kernel-modules-extra		6.1.79
kernel-modules-extra-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
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

软件包	AMI	Hyper-V VHDX
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
libffi	3.4.4	3.4.4

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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	
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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	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

软件包	AMI	Hyper-V VHDX
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-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

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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
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	

软件包	AMI	Hyper-V VHDX
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-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

软件包	AMI	Hyper-V VHDX
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
python3-libselinux	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

软件包	AMI	Hyper-V VHDX
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-policycoreutils	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
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

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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
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

软件包	AMI	Hyper-V VHDX
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
xfsprogs	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-defaults	1.1.2	
zstd	1.5.5	1.5.5

识别 Amazon Linux 实例和版本

能够确定操作系统映像或实例属于哪个 Linux 发行版及其版本可能非常重要。Amazon Linux 提供了相应的机制来将其与其他 Linux 发行版区分开，并识别映像所属的 Amazon Linux 发布版本。

这部分将介绍可用的不同方法、它们的局限性，并通过一些使用示例进行说明。

主题

使用 `os-release` 标准

Amazon Linux 遵循用于识别 Linux 发行版的 `os-release` 标准。该文件提供关于操作系统标识和版本信息的机器可读信息。

> **Note**
>
> 该标准规定首先尝试解析 `/etc/os-release`，其次是 `/usr/lib/os-release`。应注意遵循有关文件名和路径的标准。

主题

关键识别差异

`os-release` 位于 `/etc/os-release`，如果该位置不存在，则位于 `/usr/lib/os-release`。完整信息请查阅 `os-release` 标准。

确定实例是否运行 Amazon Linux 最可靠的方法是检查 `os-release` 中的 `ID` 字段。

区分不同版本最可靠的方法是检查 os-release 中的 VERSION_ID 字段：

- Amazon Linux AMI：VERSION_ID 包含基于日期的版本（例如 2018.03）
- AL2：VERSION_ID="2"
- AL2023：VERSION_ID="2023"

> **Note**
>
> 请记住，VERSION_ID 是一个供编程使用的机器可读字段，而 PRETTY_NAME 是为向用户显示而设计的。有关字段类型的更多信息，请参阅 the section called “字段类型”。

字段类型：机器可读与人类可读

/etc/os-release 文件（或者如果 /etc/os-release 不存在，则为 /usr/lib/os-release 文件）包含两种类型的字段：供编程使用的机器可读字段，以及供向用户呈现信息用的人类可读字段。

机器可读字段

这些字段使用标准化格式，旨在供脚本、程序包管理器和其他自动化工具处理。它们仅包含小写字母、数字和有限的标点符号（句点、下划线和连字符）。

- ID：操作系统标识符。Amazon Linux 在所有版本中使用 amzn，以此区别于其他发行版，如 Debian (debian)、Ubuntu (ubuntu) 或 Fedora (fedora)
- VERSION_ID：供编程使用的操作系统版本（例如 2023）
- ID_LIKE：相关发行版的空间分隔列表（例如 fedora）
- VERSION_CODENAME：供脚本使用的发布代号（例如 karoo）
- VARIANT_ID：用于编程决策的变体标识符
- BUILD_ID：系统映像的构建标识符
- IMAGE_ID：容器化环境的映像标识符
- PLATFORM_ID：平台标识符（例如 platform:al2023）

人类可读字段

这些字段旨在向用户显示，可能包含空格、混合大小写和描述性文本。在用户界面中呈现操作系统信息时应使用它们。

- NAME：用于显示的操作系统名称（例如 Amazon Linux）
- PRETTY_NAME：用于显示的包含版本的完整操作系统名称（例如 Amazon Linux 2023.8.20250721）
- VERSION：适合向用户呈现的版本信息
- VARIANT：用于显示的变体或版本名称（例如 Server Edition）

其他信息字段

这些字段提供有关操作系统的额外元数据：

- HOME_URL：项目主页 URL
- DOCUMENTATION_URL：文档 URL
- SUPPORT_URL：支持信息 URL
- BUG_REPORT_URL：错误报告 URL
- VENDOR_NAME：供应商名称
- VENDOR_URL：供应商 URL
- SUPPORT_END：支持终止日期（YYYY-MM-DD 格式）
- CPE_NAME：通用平台枚举标识符
- ANSI_COLOR：用于终端显示的 ANSI 颜色代码

当编写需要以编程方式识别 Amazon Linux 的脚本或应用程序时，请使用机器可读字段，如 ID 和 VERSION_ID。当向用户显示操作系统信息时，请使用人类可读字段，如 PRETTY_NAME。

/etc/os-release 示例

/etc/os-release 文件内容在 Amazon Linux 各版本间有所不同：

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/"
```

与其他发行版的比较

要了解 Amazon Linux 在更广泛的 Linux 生态系统中的位置，可将其 /etc/os-release 格式与其他主要发行版进行比较：

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
```

请注意机器可读字段如何提供跨发行版的一致标识：

- ID：唯一标识操作系统：Amazon Linux 为 amzn，Fedora 为 fedora，Debian 为 debian，Ubuntu 为 ubuntu
- ID_LIKE：显示发行版关系：Amazon Linux 使用 fedora（AL2023）或 centos rhel fedora（AL2），而 Ubuntu 显示 debian 以表明其源自 Debian
- VERSION_ID：提供机器可解析的版本信息：AL2023 为 2023，Fedora 为 42，Debian 为 12，Ubuntu 为 24.04

相比之下，人类可读字段专为向用户显示而设计：

- NAME：用户友好的操作系统名称：Amazon Linux、Fedora Linux、Debian GNU/Linux、Ubuntu

- PRETTY_NAME：包含版本的完整显示名称：Amazon Linux 2023.8.20250721、Fedora Linux 42 (Container Image)、Debian GNU/Linux 12 (bookworm)、Ubuntu 24.04.2 LTS
- VERSION：具有附加上下文（如代号或发布类型）的人类可读版本

编写跨平台脚本时，应始终使用机器可读字段（ID、VERSION_ID、ID_LIKE）进行逻辑判断和决策，并仅使用人类可读字段（PRETTY_NAME、NAME）向用户显示信息。

Amazon Linux 特有文件

有一些特定于 Amazon Linux 的文件可用于识别 Amazon Linux 及其版本。新代码应使用 /etc/os-release 标准以实现跨发行版兼容。不鼓励使用任何 Amazon Linux 特有文件。

主题

- /etc/system-release 文件
- 映像标识文件
- Amazon Linux 特有文件示例

/etc/system-release 文件

Amazon Linux 包含 /etc/system-release 文件，用于指定当前已安装的版本。此文件通过程序包管理器更新，在 Amazon Linux 中是 system-release 程序包的一部分。虽然 Fedora 等其他发行版也有此文件，但基于 Debian 的发行版（如 Ubuntu）中不存在。

> **Note**
>
> /etc/system-release 文件包含一个人类可读的字符串，不应以编程方式用于识别操作系统或发布版本。请改用 /etc/os-release（或者如果 /etc/os-release 不存在，则使用 /usr/lib/os-release）中的机器可读字段。

Amazon Linux 还在 /etc/system-release-cpe 文件中包含遵循通用平台枚举（CPE）规范的 /etc/system-release 的机器可读版本。

映像标识文件

每个 Amazon Linux 映像都包含一个唯一的 `/etc/image-id` 文件，该文件提供有关 Amazon Linux 团队生成的原始映像的附加信息。此文件特定于 Amazon Linux，在其他 Linux 发行版（如 Debian、Ubuntu 或 Fedora）中找不到。此文件包含有关映像的以下信息：

- `image_name`、`image_version`、`image_arch`：来自用于构建该映像的构建配方中的值。
- `image_stamp` - 映像创建期间随机生成的一个唯一的十六进制值。
- `image_date` – 映像创建的 UTC 时间，采用 YYYYMMDDhhmmss 格式。
- `recipe_name`、`recipe_id`：用于构建该映像的构建配方的名称和 ID。

Amazon Linux 特有文件示例

以下部分提供每个主要 Amazon Linux 版本的 Amazon Linux 特有标识文件示例。

> **Note**
>
> 在任何实际代码中，如果 `/etc/os-release` 文件不存在，则应使用 `/usr/lib/os-release`。

AL2023

以下示例显示 AL2023 的标识文件。

AL2023 的 `/etc/image-id` 示例：

```
[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"
```

AL2023 的 `/etc/system-release` 示例：

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

```
Amazon Linux release 2023.8.20250721 (Amazon Linux)
```

AL2

以下示例显示 AL2 的标识文件。

AL2 的 `/etc/image-id` 示例：

```
[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"
```

AL2 的 `/etc/system-release` 示例：

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

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

Amazon Linux AMI

以下示例显示 Amazon Linux AMI 的标识文件。

Amazon Linux AMI 的 `/etc/image-id` 示例：

```
[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"
```

Amazon Linux AMI 的 /etc/system-release 示例：

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

```
Amazon Linux AMI release 2018.03
```

操作系统检测示例代码

以下示例演示如何使用 /etc/os-release 文件（或者如果 /etc/os-release 不存在，则使用 /usr/lib/os-release 文件）以编程方式检测操作系统和版本。这些示例展示如何区分 Amazon Linux 与其他发行版，以及如何使用 ID_LIKE 字段来确定发行版系列。

下面的脚本以几种不同的编程语言实现，每种实现都会产生相同的输出。

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";
    }
}
```

在不同系统上运行时，该脚本将产生以下输出：

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)
```

文件系统布局

这部分涵盖 AL2023 系统的文件系统布局，包括可能特定于实例或基于 AL2023 的容器的详细信息。更多信息请参阅 `file-hierarchy(7)` man 页面。

主题

/ (根目录)

默认情况下，AL2023 映像配置为具有可写的 `/`，允许特权用户创建新文件和目录。

可以配置 `systemd` 服务以使用不同的路径或映像作为该服务的 `/`，并对任何路径设置访问限制。

> **Note**
>
> 最佳实践是将 `systemd` 服务配置为限制该服务可以访问的内容。这可能包括使用 `ReadOnlyPaths=/` 指令，使 `/` 对该服务只读。
> 有关使用 `systemd` 限制服务对系统访问权限的更多信息，请参阅 `systemd.exec(5)` man 页面。

/boot (内核、initramfs 等)

默认情况下，可引导的 AL2023 映像配置为将 `/boot` 放在 root 文件系统上。`/boot` 路径仅与可引导映像相关，因此在 AL2023 容器映像中未使用。

此目录包含 AL2023 启动所需的文件，例如 Linux 内核和 initramfs。此目录的内容应仅使用操作系统提供的工具进行操作。

/boot/efi（EFI 系统分区）

默认情况下，可引导的 AL2023 映像配置为将 EFI System 分区挂载在 /boot/efi。此文件系统由操作系统管理，包含对系统启动至关重要的代码和配置。

此路径与容器映像无关。

/etc（系统配置）

AL2023 上的 /etc 目录包含特定于系统的配置。默认情况下，AL2023 映像将 /etc 放在 root 文件系统上，并且可由特权用户写入。

> **Note**
>
> 应用程序（包括 systemd）通常将默认配置保留在 /usr（系统资源）下，可以通过将配置放在 /etc（系统配置）中来覆盖这些默认配置。
> 对于这些应用程序，更改 /usr（系统资源）中的文件而不是覆盖 /etc 中的默认配置，可能会导致在程序包更新时更改被覆盖。

/home（用户主目录）

普通用户的主目录位于 /home 下，但软件应始终查找每个用户的 $HOME 环境变量，而不是依赖诸如 /home/$USER 之类的模式。

默认情况下，AL2023 映像将 /home 放在 root 文件系统上，但软件不应依赖于此。将操作系统配置为 /home 是一个独立的文件系统是完全有效的，该文件系统可能在启动过程中稍后挂载，或者仅在用户通过系统身份验证后才挂载。

root 用户的主目录不在 /home 中，而是在 /root（root 用户主目录）中，以便在无法挂载 /home 文件系统时仍然可用。

> **Note**
>
> 对于不需要写入访问 /home 的 systemd 服务，最佳实践是配置 ProtectHome=read-only 指令。使用此选项，/home、/root 和 /run/user 对该服务将变为只读。

同样，对于不需要任何访问 /home 的服务，最佳实践是配置 ProtectHome=tmpfs 指令，这将在沙盒中运行该服务，其中 /home、/root 和 /run/user 是空的只读 tmpfs 文件系统。有关使用 systemd 限制服务对系统访问权限的更多信息，请参阅 systemd.exec(5) man 页面。

/root（root 用户主目录）

root 用户的主目录是 /root 目录，特意与 /home（用户主目录）分开，以便在 /home（用户主目录）位于不可用的文件系统上时，该目录仍然存在。

配置 systemd 服务的最佳实践对于 /root 和 /home（用户主目录）是相同的。

/srv（服务器有效载荷）

/srv 目录由系统管理员管理，Amazon Linux 2023 对此目录的组织方式没有任何限制。

可以将 /srv 目录配置为位于单独的文件系统上，因此它可能仅在启动过程的后期才可用。

/tmp（小型临时文件）

Note

Amazon Linux 2023 与 Amazon Linux 2 不同，默认情况下 /tmp 现在是 tmpfs，而不是 root 文件系统上的一个路径。

Note

在容器中运行时，通常由您的容器运行时配置决定 /tmp 是 tmpfs 还是磁盘上的路径，以及是否有运行的清理进程。

/tmp 目录用于存放小型、大小受限的临时文件。默认情况下，AL2023 将其配置为 tmpfs 文件系统，大小限制为 RAM 的 50%，且最多一百万个 inodes。

应用程序应优先使用 $TMPDIR 环境变量中的路径，而非 /tmp。然后用户可以设置 $TMPDIR 环境变量来覆盖应用程序应用于 /tmp 的路径

对于较大的临时文件，应改用 /var/tmp。

> **Warning**
>
> 由于 /tmp 是共享的，因此使用安全的方法创建临时文件非常重要。有关详细信息，请参阅上游关于安全使用 /tmp 和 /var/tmp 的 systemd 文档。

> **Note**
>
> 最佳实践是为 systemd 服务配置 PrivateTmp= 指令设置为 yes 或 disconnected，这将在沙盒中运行服务，其中 /tmp 和 /var/tmp 不与主机或其他服务共享。
> 更多信息，包括如何配置两个服务以共享相同的私有临时目录，请参阅 systemd.exec(5) man 页面。

/tmp 的内容通常在启动时被清理，未使用的文件会定期清理。默认情况下，清理过程在启动后不久运行，然后每天运行。有关如何配置临时文件清理的信息，请参阅 tmpfiles.d(5) 和 systemd-tmpfiles(8) 的 man 手册页。

/tmp 和 /var/tmp 路径密切相关，但存在的目的不同。

/run（运行时数据）

/run 目录被系统程序包用于存储少量运行时数据（例如套接字文件）。它是一个 tmpfs 文件系统，并且仅可由特权程序写入。

/run/log 目录可被系统组件用来存储日志，无论是在写入 /var/log 之前，还是在 /var/log 文件系统可用之前。

/run/user/ 路径包含每个用户的运行时目录。默认情况下，这些将是独立的 tmpfs 文件系统，在用户登录时由 systemd 挂载，并在用户不再登录时被擦除。根据 XDG 基础目录规范，不应直接引用这些路径，而应通过 $XDG_RUNTIME_DIR 环境变量引用。

/usr（系统资源）

/usr 层级结构用于供应商提供的操作系统资源。除了 /usr/local 层级结构外，任何东西（操作系统程序包管理器除外）都不应修改 /usr 下的任何内容。

软件应用程序必须假定 /usr 可以是只读的。/usr 层级结构不得用于易失性数据。除了 /usr/local 之外，/usr 层级结构不得用于任何在操作系统程序包管理器执行程序包安装/移除之外添加或更改的数据。操作系统程序包管理器可以假定整个 /usr 层级结构（/usr/local 除外）是同一个挂载点。

在操作系统程序包管理器之外安装的软件不应将数据存储在 /usr 中，因为这可能会妨碍未来任何对操作系统程序包管理器的调用。/usr/local 层级结构是个例外，它保留给操作系统程序包管理器之外的软件使用。

/usr/bin（可执行文件）

应出现在标准搜索 $PATH 中的可执行文件，并且从 shell 调用很有用。从 shell 调用无用的守护进程和可执行文件则存放在 /usr/lib 或 /usr/libexec 中。

/usr/include（C/C++ 头文件）

/usr/include 目录包含 C 和 C++ 头文件，通常包含在带有 -devel 后缀的程序包中。

/usr/lib 和 /usr/lib64（共享库）

在 Amazon Linux 2023 上，/usr/lib64 路径用于 64 位共享库以及依赖于架构的程序包数据。由于 AL2023 不附带任何 32 位用户空间支持，因此只有 64 位共享库可用。

/usr/lib 路径用于来自操作系统程序包且与所有架构兼容的静态数据。这可能包括通常不从 shell 调用的可执行文件，这些文件也可能在 /usr/libexec 中找到。共享库位于 /usr/lib64 而非 /usr/lib。

/usr/local（系统管理员安装的软件）

在 Amazon Linux 2023 上，/usr/local 路径可供系统管理员安装不属于该操作系统且不会被操作系统触及的软件。默认的 /usr/local 层级结构镜像了 / 层级结构。

/usr/share（共享资源）

共享资源（如文档、字体和时区数据）存放在 /usr/share 中。通常有各种规范明确规定数据在此目录中的存储位置和格式。

/usr/share/doc（文档）

随程序包附带的文档将存储在 /usr/share/doc 中。

/var（持久性变量系统数据）

/var/cache（缓存）

与 /var/lib 不同，删除 /var/cache 中的数据不会导致数据丢失，因为要求应用程序能够从其他来源重建其 /var/cache 数据。

/var/lib（持久性系统数据）

/var/lib 目录用于持久性系统数据。各种系统组件会将其私有数据放置于此。与 /var/cache 不同，删除 /var/lib 中的数据将导致数据丢失。

例如，PostgreSQL 数据库服务器默认将数据库数据存储在 /var/lib/pgsql 中。此数据的布局和文件格式是 PostgreSQL 私有的，并且是持久性数据，因为如果被擦除，用户将经历数据丢失。

/var/log（持久性日志）

此目录用于存储持久性日志。建议软件使用 syslog(3) 或 sd_journal_print(3) API 调用，而不是直接在 /var/log 下存储日志文件。

> **Note**
>
> 在 AL2023 中，systemd 日志取代 rsyslog，这与默认的 Amazon Linux 2 配置有显著差异。

有关使用 journalctl 读取日志的更多信息，请参阅 journalctl 手册页。

许多应用程序使用自己的机制来写入（有时还会轮换）位于 /var/log 中的日志文件。有关如何配置其日志文件，请参阅这些应用程序的文档。

/var/spool（邮件和打印机队列）

此目录用于存储持久性数据，例如邮件或打印队列。

/var/tmp（较大的临时文件）

对于小型、大小受限的临时文件，可能应改用 /tmp。

虽然 /tmp 默认配置为 tmpfs 卷，但 /var/tmp 默认配置为根文件系统上的一个路径，因此是存放较大且更持久临时文件的地方。默认情况下，会定期运行清理作业，删除最近未访问的文件。

有关如何配置临时文件清理的信息，请参阅 tmpfiles.d(5) 和 systemd-tmpfiles(8) 的 man 手册页。

与 /tmp 一样，应用程序应优先使用 $TMPDIR 环境变量中指定的路径，而非 /var/tmp。然后用户可以设置 $TMPDIR 环境变量来覆盖应用程序应用于 /var/tmp 的路径。

> **Warning**
>
> 由于 /var/tmp 是共享的（/tmp 也是如此），因此使用安全的方法创建临时文件非常重要。有关详细信息，请参阅上游关于安全使用 /tmp 和 /var/tmp 的 systemd 文档。

> **Note**
>
> 最佳实践是为 systemd 服务配置 PrivateTmp= 指令设置为 yes 或 disconnected，这将在沙盒中运行服务，其中 /tmp 和 /var/tmp 不与主机或其他服务共享。
> 更多信息，包括如何配置两个服务以共享相同的私有临时目录，请参阅 systemd.exec(5) man 页面。

/tmp 和 /var/tmp 路径密切相关，但存在的目的不同。

正在更新 AL2023

重要的是要及时了解最新 AL2023 版本，这样您才能从安全更新和新功能中受益。使用 AL2023，您可以通过确保整个环境中的软件包版本和更新之间的一致性通过版本控制的存储库进行确定性升级 AL2023。

> **Warning**
>
> 运行 `dnf --releasever=latest update` 并非最佳实践，很可能导致操作系统更新首先在生产环境中进行测试。
> 不使用`latest`，而是使用特定的 AL2023 发行版本。这确保您在生产实例上部署的更改与先前测试的更改一致。例如，`dnf --releasever=2023.10.20260216 update`将始终更新到 2023.10.20260216 版本。
> 有关更多信息，请参阅《AL2023 用户指南》中的 AL2023 “更新” 部分。

主题

安全部署更新的最佳实践

Amazon Linux 2023 (AL2023) 具有多项功能，旨在帮助安全地将更新部署到操作系统，并能够知道两次更新之间发生了哪些变化，并在必要时轻松恢复到旧版本。本节探讨了 AWS 从十多年内部和外部使用 Amazon Linux 中吸取的经验教训。

> **Warning**
>
> 运行 `dnf --releasever=latest update` 并非最佳实践，很可能导致操作系统更新首先在生产环境中进行测试。

不使用`latest`，而是使用特定的 AL2023 发行版本。这确保您在生产实例上部署的更改与先前测试的更改一致。例如，`dnf --releasever=2023.10.20260216 update`将始终更新到 2023.10.20260216 版本。
有关更多信息，请参阅《AL2023 用户指南》中的 AL2023 “更新” 部分。

如果不对操作系统更新的部署安全性进行规划，则操作系统更新 application/service 与操作系统更新之间意外负面交互的影响可能会大得多，最高可达并包括完全中断。与任何软件问题一样，问题发现得越早，对最终用户的影响就越小。

重要的是不要陷入两个根本错误的认知陷阱：

1. 操作系统供应商在操作系统更新中永远不会犯错。
2. 您所依赖的操作系统的特定行为或接口，与操作系统供应商认为应被依赖的行为和接口一致。

   即操作系统供应商和您都认同更新可能存在问题。

不要依赖良好意愿，应建立系统确保部署安全性包含任何操作系统更新。

不建议通过部署到生产环境来测试新的操作系统更新。最佳实践是将操作系统视为部署的另一组成部分，并考虑应用您认为适用于生产环境任何其他变更的相同部署安全机制。

最佳实践是在部署到生产系统之前测试所有操作系统更新。部署时，建议采用分阶段发布并结合良好的监控。分阶段发布可以确保即使出现问题（即使不是立即出现），影响也仅限于实例集的一部分，同时可以在进行进一步调查和缓解时停止继续部署更新。

减轻操作系统更新带来的负面影响通常是首要任务，随后才是在任何可能出现问题的地方解决问题。当操作系统更新的引入与负面影响相关联时，能够回退到先前已知正常的操作系统版本是一个强大的工具。

Amazon Linux 2023 引入了 通过版本化存储库实现确定性升级，这是一个强大的新功能，旨在确保对操作系统（或单个程序包）版本的任何更改都是可重复的。因此，如果在从一个操作系统版本迁移到下一个版本时遇到问题，可以使用简单易用的机制来坚持使用已知正常工作的操作系统版本，同时找出解决问题的方法。

因此 AL2023，每当我们发布新的软件包更新时，都会有一个新版本需要锁定，而新 AMIs 版本则锁定到该版本。AL2023 发行说明涵盖了每个版本中的更改，并亚马逊 Linux 安全公告 AL2023涵盖了软件包更新中解决的安全问题。

例如，如果您受到 2023.6. 20241028 版本中存在的问题的影响，则可以立即恢复使用先前版本 2023.6.20241010 的 AMIs 和容器镜像。在这种情况下，某个程序包中存在一个错误，该错误在随后的 2023.6.20241031 发布版本中得到了修复，但借助 通过版本化存储库实现确定性升级，任何受影响的人都可以立即采取简单措施来缓解：只需使用先前的映像即可。

通过版本化存储库实现确定性升级还可确保任何正在进行的操作系统更新部署，无论是就地部署还是通过启动新的映像 AMIs 或容器映像，都不会受到随后发布的操作系统更新的影响。

对于我们的第一个示例，实例集 A 是一个大型实例集，在部署从 2023.5.20241001 到 2023.6.20241010 发布版本的更新过程进行到一半时，2023.6.20241028 发布版本发布了。通过版本化存储库实现确定性升级 意味着实例集 A 的部署将继续进行，其应用的更新内容不会有任何改变。

基于波浪或分阶段的部署策略（例如首先部署到实例集的 1%，然后是 5%、10%、20%、40%，直到达到 100%）的目的，是能够在更广泛推广之前以有限的方式测试变更。此类部署策略通常被认为是部署任何生产变更的最佳实践。

采用波浪式部署策略，且实例集 A 更新至 2023.6.20241010 的阶段正处于同时部署到大量主机时，得益于使用 通过版本化存储库实现确定性升级，2023.6.20241028 的发布对正在进行的部署没有影响。

如果实例集 B 运行的是较旧版本（例如 2023.5.20240708），并已开始将更新部署至 2023.6.20241028，且实例集 B 受到该版本中问题的影响，这将在部署早期被发现。此时，可以决定是暂停任何推广，直到该问题的修复可用，还是在此期间开始部署实例集 A 正在运行的相同版本 2023.6.20241010，以便实例集 B 获得 2023.5.20240708 到 2023.6.20241010 之间的所有更新。

务必注意，未能及时进行操作系统更新可能会导致问题。新更新可能包含与您环境相关的错误修复和安全修复。有关更多信息，请参阅Amazon Linux 2023 中的安全性与合规性和在中管理软件包和操作系统更新 AL2023。

配置部署系统使其能够轻松获取新的操作系统更新，在部署到生产环境之前进行测试，并使用波浪式部署等机制来最小化任何负面影响，这一点非常重要。为了能够减轻操作系统更新的任何负面影响，重要的是要知道如何让您的部署系统指向先前已知正常的操作系统版本，并且在问题解决后，不再锁定到旧的已知正常版本，而是迁移到新的已知正常版本。

为次要更新做准备

为操作系统的较小更新（例如新的单点版本）做准备的目的 AL2023仅限于零努力。请务必阅读AL2023 发行说明，了解即将发生的任何变化。

程序包的支持周期结束可能涉及迁移到较新版本的语言运行时（例如 PHP在 AL2 023）。最佳实践是提前规划，在支持周期结束前从容地迁移到新的语言运行时版本。

对于像 pcre 版本 1 这样的程序包，也有机会提前规划并将您的任何代码迁移到其替代品，在此例中是 pcre 版本 2。最佳实践是尽快这样做，以便为任何意外情况留出时间。

在没有直接替代品的情况下，例如 Berkeley DB (libdb)，您可能需要根据您的使用案例做出选择。

为主要更新做准备

更新到操作系统的新主要版本几乎普遍被认为需要规划、需要投入工作以适应已更改或已弃用的功能，并且在部署前需要进行测试。通常可以更渐进式地为下一个 Amazon Linux 2023 主要版本做准备，例如在进行下一个主要版本迁移之前，先解决任何对已弃用或已移除功能的使用问题。

例如，当从移动 AL2 到时 AL2023，阅读该已在中弃用 AL2 和删除的功能 AL2023部分可能会导致许多安全而小的步骤，这些步骤可能会在仍在 AL2 使用时发生 AL2023。例如，任何 Python 2.7 已被 Python 3 所取代 的使用（操作系统使用之外的情况，例如在 yum 程序包管理器中）都可以迁移到 Python 3，为使用 AL2023 中的 Python 做准备。如果使用 PHP，则两者 AL2（通过 PHP 8.2 E AL2 x tra）和 PHP 8.2 AL2023 发布，因此 PHP 版本迁移和操作系统迁移不必同时进行。

在使用时 AL2023，还可以在使用的同时为当今亚马逊 Linux 2023 的下一个主要版本做好准备 AL2023。本AL2023 中已弃用的功能节介绍已弃用和即将移除的功能 AL2023 和软件包。

例如，将任何剩余的 System V init (sysvinit) 使用（例如 init 脚本）迁移到其 systemd 等效项，将使您为未来做好准备，同时也允许您使用完整的 systemd 功能集来监控服务、决定如何以及是否重启它、它需要哪些其他服务，以及是否应该应用任何资源或权限约束。

对于诸如 32 位支持之类的功能，弃用过程可能跨越操作系统的多个主要版本。对于 32 位，亚马逊 Linux 1 (AL1) 已弃用，32 位 x86（i686）AMI亚马逊 Linux 2 已弃用，亚马逊 Linux 32 位 x86（i686）程序包 2023 已弃用。32 位 x86（i686）运行时支持从 IMDSv1 的过渡也跨越了操作系统的多个主要版本。对于这些类型的变更，我们理解部分客户需要更长的时间来适应，因此在 Amazon Linux 2023 中该功能完全不可用之前，会有很大的宽限期。

已弃用功能列表会在操作系统的生命周期内更新，建议随时关注其变更。

有新的更新时收到通知

您可以在新的 AL2023 AMI 发布时接收通知。通过 Amazon SNS 使用以下主题发布通知。

```
arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates
```

当有新的 AL2023 AMI 发布时，将在此处张贴消息。AMI 的版本将包含在消息中。

可以使用几种不同的方法接收这些消息。我们建议您使用以下方法。

1. 打开 Amazon SNS 控制台。
2. 在导航栏中，可根据需要将 AWS 区域改为美国东部（弗吉尼亚州北部）。必须选择创建您订阅的 SNS 通知的区域。
3. 在导航窗格中，依次选择订阅、创建订阅。
4. 对于 Create subscription 对话框，执行以下操作：
   a. 对于主题 ARN，请复制并粘贴以下 Amazon 资源名称（ARN）：**`arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates`**。
   b. 对于协议，选择电子邮件。
   c. 对于端点，输入可用来接收通知的电子邮件地址。
   d. 选择创建订阅。
5. 您将收到一封主题行为“AWS 通知 - 订阅确认”的确认电子邮件。打开该电子邮件，选择确认订阅来完成订阅。

通过版本控制的存储库进行确定性升级 AL2023

> **Note**
>
> 默认情况下，您的 AL2023 实例在启动时不会自动收到其他关键和重要的安全更新。您的实例最初包含的版本 AL2023 和所选 AMI 中可用的更新。

控制从主要版本和次要版本收到的更新

使用 AL2023，您可以确保整个环境中的软件包版本和更新之间的一致性。您还可以确保同一个 Amazon 机器映像 (AMI) 的多个实例的一致性。利用通过版本控制的存储库的确定性升级功能（默认开启），您可以根据满足您特定需求的时间表来应用更新。

每当我们发布新的软件包更新时，都会有一个新版本需要锁定，而新 AMIs 版本则锁定到该版本。

AL2023 锁定到存储库的特定版本。主要版本和次要版本都支持此功能。通过我们的 SSM 参数公开的 AL2023 AMI 始终是最新版本。它包含最多的 up-to-date软件包和更新，包括关键和重要的安全更新。

如果您从一个现有 AMI 启动实例，则不会自动应用更新。作为配置的一部分安装的任何其他软件包映射到现有 AMI 的存储库版本。

借助此功能，您负责确保环境中程序包版本和更新的一致性。当您从同一个 AMI 启动多个实例时，则功能尤其有用。您可以根据满足您需求的时间表应用更新。您还可以在启动时应用一组特定的更新，因为这些更新也可以锁定到特定的存储库版本。

主要版本升级和次要版本升级之间的区别

的主要版本 AL2023 包括大规模更新，可能会添加、删除或更新软件包。在将实例升级到某个新的主要版本时，为确保兼容性，应先在该版本上测试了您的应用程序后再升级。

的次要版本 AL2023 包括功能和安全更新，但不包括软件包更改。这确保了 Linux 功能和系统库 API 在新版本中保持可用。在更新前测试您的应用程序并非必需。

了解更新何时可用

为了应用更新，您需要知道有更新可用，然后了解如何部署该更新。

对于在新 AL2023 AMIs 版本发布 AMIs 时派生的构建，EC2 Image Builder 可以自动构建、修补和测试 AMIs。要触发自己的 AMI 构建管道或使用基础 AMIs，您可以有新的更新时收到通知。

对于原地修补，您可以使用诸如 AWS Systems Manager Patch Manager 之类的工具来协调跨实例集的应用更新。

对于其他 AMIs 基于的公众 AL2023，其提供者 AMIs 可能有自己的发布时间表和通知方式。使用派生镜像 AMIs 或容器镜像时，请查看发行商提供的文档，了解何时发布更新。

每个版本的更改都记录在AL2023 发行说明中。安全更新发布在 Amazon Linux 安全中心（ALAS）。

控制 AL2023 存储库中可用的软件包更新

当我们发布新版本的 AL2023 存储库时，所有以前的版本仍然可用。默认情况下，用于管理存储库版本的插件锁定到用于构建 AMI 的相同版本。如果要控制软件包更新，请按照下列步骤操作。

1. 通过运行以下命令发现可用的存储库版本。

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

2. 您可以通过运行以下命令选择一个版本。

```
$ sudo dnf upgrade --releasever=version
```

该命令使用 dnf 启动一个更新，从您当前的 Amazon Linux 发行版本更新为命令行中指定的发行版本。dnf 将显示一个软件包更新列表。在处理更新之前，您必须先确认更新。更新完成后，新的发行版本变成 dnf 用于所有未来活动的默认发行版本。

有关更多信息，请参阅 在中管理软件包和操作系统更新 AL2023。

通过实例替换实现确定性更新

Amazon Linux 2023 的 通过版本控制的存储库进行确定性升级 AL2023 功能使得实例替换成为一种简单、确定且安全地推出 AL2023 更新版本的方法。确定性更新意味着随着新版本的逐步推出，如果发现任何问题，可以在确定问题原因的同时轻松回退到之前的 AMI。

使用实例替换而非就地修补意味着更新更具确定性和可预测性，因为启动新容量可以是一个经过充分测试的代码路径，具有清晰的 A 和 B 状态。在部署开始之前，可以在 CI/CD 系统中充分测试更新前和更新后的每种状态。

在进行就地修补时，应用更新前后的中间状态很多，很难测试所有状态组合。

采用具有确定性更新的实例替换这一操作系统更新策略，非常适用于蓝/绿、分波和分阶段部署模型。

使用版本化存储库实现确定性升级

主题

使用确定性升级的系统

> **Note**
>
> 包管理器的默认行为已从改为 AL2。

确定性升级是一种有效的方法，可确保对生产环境的所有更改在广泛部署之前得到充分测试。每个新 AL2023 AMI 都锁定到特定版本的 AL2023。这提供了启动特定 AMI 时将安装的操作系统程序包版本的确定性行为。原地更新可以升级到特定的发布版本，确保整个实例集中的确定性行为。当你迁移到新的

AMIs 或就地更新版本时，你可以测试 CI/CD 管道中的每个版本，在部署到生产环境之前发现任何潜在的问题。

您可以使用诸如 AWS Systems Manager Patch Manager 之类的工具来协调跨实例集的应用更新。对于在新 AL2023 AMIs 版本发布 AMIs 时派生的构建，EC2 Image Builder 可以自动构建 AMIs、修补和测试，或者您可以有新的更新时收到通知知道新基础何时可用，或者触发自己 AMIs 的 AMI 构建管道。

有关将更新限制为来自特定通告的信息，请参阅 原地应用安全更新

对于原地修补，您可以使用 dnf 程序包管理器。但您运行 dnf upgrade 命令时，系统会检查 releasever 变量指定的存储库中的升级。有效版本releasever是其中一个*latest*或一个带有日期戳的版本，例如. *2023.10.20260216*

您可以使用以下任一方法更改 releasever 的值。这些方法按系统优先级降序列出。这意味着方法 1 优先于方法 2 和 3，方法 2 优先于方法 3。

1. 命令行标志中的值，--releasever=*latest*（如果使用）。
2. 替代变量文件中指定的值，/etc/dnf/vars/releasever（如果设置）。
3. 当前安装的 system-release 软件包版本。

在以下示例中，版本为*2023.0.20230210*：

```
$ rpm -q system-release
system-release-2023.0.20230210-0.amzn2023.noarch
```

在新安装的系统中，替代变量不存在。由于系统锁定到 system-release 的已安装版本，所以无升级可用。

```
$ 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!
```

要获取某个特定版本的软件包，您可以通过使用 releasever 标志来提供所需的版本。

```
$ 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
```

由于 `--releasever` 选项优先于 `system-release` 和 `/etc/dnf/vars/releasever`，所以此升级的结果如下：

1. 升级将替换在先前版本和新版本之间有所变化的所有已安装的软件包。
2. 升级会将系统锁定到 `system-release` 的新版本的存储库。

通过始终指定要更新的内容`releasever`（即 AL2023 版本），您可以对队列进行一组确定性的更改。您启动了版本*A*，更新为*B*，然后更新为*C*。

选择性更新确定性升级系统

> **Note**
>
> 我们建议安装新发布版本中的所有更新，而不是选择特定更新。仅对操作系统应用部分更新，应被视为偏离采用完整更新的标准实践的例外情况。

您或许想要安装来自最近版本的选定软件包，同时仍将系统锁定到原始发行版本。

您可以使用 `dnf check-update` 来确认要升级的软件包。

```
$ 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
```

安装您要升级的程序包。使用 `sudo dnf upgrade --releasever=latest` 和软件包名称来确保 `system-release` 软件包保持不变。

```
$ 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**
>
> 使用 `sudo dnf upgrade --releasever=latest` 更新所有软件包，包括 `system-release`。然后，该版本仍锁定到新的 `system-release`，除非您设置了持久取代。

在确定性升级中使用持久取代

> **Note**
>
> 通过确定性更新，您可以将操作系统更改集成到您的 CI/CD 管道中。禁用确定性更新将丧失在部署前进行测试的能力。

您可以通过将变量值设置为`--releasever=latest`，使用永久覆盖来解锁系统，而不必添加*`latest`*。通过始终使用`latest`，这会将的行为恢复 AL2023 到 AL2 更新模型，在该模型中，对包管理器的任何调用都将始终查看最新版本，并且不会锁定到任何特定版本的操作系统。

> **Warning**
>
> 通过使用对确定性更新的持久覆盖来解锁程序包管理器，您将承担在生产环境中发现您的应用程序与操作系统更新之间任何可能的不兼容性的风险。
> 虽然不兼容性确实罕见，但通过操作系统更新，您正在将新的代码变更集成到您的环境中，集成测试可以防止部署对生产环境产生负面影响的代码变更。

```
$ 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**
>
> 如果您使用了取代变量 `/etc/dnf/vars/releasever`，请使用以下命令，通过擦除取代值来恢复默认锁定行为。
>
> ```
> $ sudo rm /etc/dnf/vars/releasever
> ```

使用永久替换使用`latest`而不是特定版本类似于的默认行为。 AL2有些服务是 AMIs基于这些服务来构建的， AL2 它们会禁用此行为，并锁定到特定的软件包版本，就像你默认开启的那样 AL2023。

我们建议使用从新 AMI 启动的实例替换原有实例，而不是禁用确定性更新。如果无法进行实例替换，我们建议使用诸如 AWS Systems Manager Patch Manager 之类的工具来协调跨实例集的应用更新。EC2 Image Builder 还可以自动构建、修补和测试 AMIs 您自己的 AL2023 基础映像。您也可以 有新的更新时收到通知，这可用于触发您自己的 AMI 构建流水线。

在预生产环境中使用 `latest`，然后使用 `latest` 部署到生产环境，并不能防范操作系统更新与您的应用程序之间的任何问题。新 AL2023 版本可能在任何时间点发布，因此在生产`latest`中的所有使用都存在风险。

在中管理软件包和操作系统更新 AL2023

与之前版本的亚马逊 Linux 不同，它 AL2023 AMIs 被锁定到特定版本的亚马逊 Linux 存储库。要对 AL2023 实例应用安全修复和错误修复，请将DNF配置更新到最新的可用发行版本。或者，启动一个较新的 AL2023 实例。

本部分介绍如何在运行的实例上管理 DNF 软件包和存储库。另外，还将介绍如何根据用户数据脚本来配置 DNF，以在启动时启用最新可用的 Amazon Linux 存储库。有关更多信息，请参阅 DNF 命令参考。

建议应用新 AL2023版本中的所有可用更新。仅选择安全更新或仅特定更新应是例外而非规则。要列出哪些 安全通告 与特定实例相关，请参阅 列出适用的通告。有关安装仅与特定通告相关的更新的信息，请参阅 原地应用安全更新。

> **Important**
>
> 如果您想报告漏洞或对 AWS 云服务或开源项目有安全疑虑，请使用 “漏洞报告” 页面与 AWS 安全部门联系

主题

查看可用的软件包更新

您可以使用 `dnf check-update` 命令查看有关您的系统的任何更新。对于 AL2023，我们建议您在命令中添加该`--releasever=`*`version-number`*选项。

添加此选项后，DNF 还会查看有关更高版本的存储库的更新。例如，当您运行 `dnf check-update` 命令后，使用最新返回的版本作为 *`version-number`* 的值。

如果实例更新为使用存储库的最新版本，输出将包括所有要更新的程序包列表。

> **Note**
>
> 如果您没有在 `dnf check-update` 命令中使用可选标志来指定发布版本，则只查看当前配置的存储库版本。这意味着不会查看存储库的更高版本中的软件包。

Updates in a specific version

在此示例中，我们将查看如果启动了一个 2023.0.20230315 发布的容器，在 2023.1.20230628 发布中有哪些更新可用。

> **Note**
>
> 此示例使用 2023.0.20230315 和 2023.1.20230628 版本，它们不是最新版本的最新版本。有关最新版本，AL2023请参阅包含最新安全更新的发行说明。AL2023

在此示例中，我们将从 2023.0.20230315 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 `.0` 表示特定发布的映像版本；此映像版本通常为零。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中检查更新。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
   bash-5.2#
```

现在使用 `dnf check-update` 命令检查 2023.1.20230628 发布中可用的更新。

> **Note**
>
> 应用程序包更新是一项特权操作。尽管在容器中运行时通常不需要提升特权，但如果在非容器化环境（例如 Amazon EC2 实例）中运行，您可以在不提升特权的情况下检查更新。

```
$ 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#
```

`system-release` 程序包的版本显示了 `dnf upgrade` 命令将更新到的发布，即 `dnf check-update --releasever=2023.1.20230628` 命令中请求的 2023.1.20230628 发布。

Updates in the latest version

在这个例子中，我们将看看 AL2023 如果我们启动了 2023.4.20 2403 `latest` 19 版本的容器，那么版本中有哪些可用的更新。在撰写本文时，`latest` 发布为 2023.5.20240708，因此此示例中列出的更新将截至该发布。

> **Note**
>
> 此示例使用 2023.4.20240319 和 2023.5.20240708 发布，后者是在撰写本文时的最新发布。有关最新版本的更多信息，请参阅AL2023 发行说明。

在此示例中，我们将从 2023.4.20240319 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 `.1` 表示特定发布的映像版本。虽然映像版本通常为零，但此示例使用的发布的映像版本为一。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中检查更新。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
   bash-5.2#
```

现在使用 `dnf check-update` 命令检查 `latest` 发布中可用的更新，该发布在撰写本文时为 2023.5.20240708。

> **Note**
>
> 应用程序包更新是一项特权操作。尽管在容器中运行时通常不需要提升特权，但如果在非容器化环境（例如 Amazon EC2 实例）中运行，您可以在不提升特权的情况下检查更新。

```
$ 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#
```

`system-release` 程序包的版本显示了 `dnf upgrade` 命令将更新到的发布。

对于此命令，如果有更新的软件包可用，则返回码为 100。如果没有更新的软件包可用，则返回码为 0。此外，输出中还会列出所有要更新的软件包。

使用 DNF 和存储库版本应用安全更新

新的软件包更新和安全更新仅适用于新的存储库版本。对于从早期 AL2023 AMI 版本启动的实例，必须先更新存储库版本，然后才能安装安全更新。`dnf check-release-update` 命令包括一个示例更新命令，可将系统上安装的所有软件包更新为较新存储库中的版本。

> **Note**
>
> 如果您没有在 `dnf check-update` 命令中使用可选标志来指定发布版本，则只查看当前配置的存储库版本。这意味着存储库任何后续版本中对已安装程序包的任何更新都不会被应用。

这部分涵盖推荐的升级路径，即应用所有可用更新，而不是挑选单个更新或仅标记为安全更新的更新。通过应用所有更新，现有实例将迁移至与启动更新的 AMI 相同的程序包集。这种一致性减少了整个实例集中程序包版本的差异。有关应用特定更新的更多信息，请参阅 原地应用安全更新。

Applying updates in a specific version

在此示例中，如果启动了一个 2023.0.20230315 发布的容器，我们将应用 2023.1.20230628 发布中可用的更新。

> **Note**
>
> 此示例使用 2023.0.20230315 和 2023.1.20230628 版本，它们不是最新版本的最新版本。有关最新版本， AL2023请参阅包含最新安全更新的发行说明。AL2023

在此示例中，我们将从 2023.0.20230315 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 `.0` 表示特定发布的映像版本；此映像版本通常为零。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中应用更新。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
   bash-5.2#
```

现在使用 `dnf upgrade` 命令应用 2023.1.20230628 发布中包含的所有更新。

> **Note**
>
> 应用程序包更新是一项特权操作。尽管在容器中运行时通常不需要提升特权，但如果在非容器化环境（例如 Amazon EC2 实例）中运行，您将需要以 `root` 用户身份运行 `dnf upgrade` 命令。这可以使用 `sudo` 或 `su` 命令来完成。

```
$ 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]:
```

system-release 程序包的版本显示了 dnf upgrade 命令将更新到的发布，即 dnf upgrade --releasever=*2023.1.20230628* 命令中请求的 2023.1.20230628 发布。

默认情况下，dnf 会要求您确认是否希望应用更新。您可以通过使用 dnf 的 -y 标志来绕过此提示。在此示例中，dnf upgrade -y --releasever=*2023.1.20230628* 命令在应用更新前不会请求确认。这在脚本或其他自动化环境中非常有用。

一旦确认要应用更新，dnf 就会应用它们。

```
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

在此示例中， AL2023 如果我们启动了 2023.4.20 2403 `latest` 19 版本的容器，我们将应用版本中可用的更新。在撰写本文时，`latest` 发布为 2023.5.20240708，因此此示例中列出的更新将截至该发布。

> **Note**
>
> 此示例使用 2023.4.20240319 和 2023.5.20240708 发布，后者是在撰写本文时的最新发布。有关最新版本的更多信息，请参阅AL2023 发行说明。

在此示例中，我们将从 2023.4.20240319 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 `.1` 表示特定发布的映像版本。虽然映像版本通常为零，但此示例使用的发布的映像版本为一。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中应用更新。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
   bash-5.2#
```

现在使用 `dnf upgrade` 命令应用 `latest` 发布中可用的更新，该发布在撰写本文时为 2023.5.20240708。

> **Note**
>
> 应用程序包更新是一项特权操作。尽管在容器中运行时通常不需要提升特权，但如果在非容器化环境（例如 Amazon EC2 实例）中运行，您将需要以 root 用户身份运行 dnf upgrade 命令。这可以使用 sudo 或 su 命令来完成。

默认情况下，dnf 会要求您确认是否希望应用更新。在此示例中，我们通过使用 dnf 的 -y 标志来绕过此提示。

```
$ 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.rp 4.1 MB/s | 105 kB     00:00
(10/21): krb5-libs-1.21-3.amzn2023.0.4.x86_64.rp  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.rp 1.1 MB/s |  26 kB     00:00
(16/21): python3-3.9.16-1.amzn2023.0.8.x86_64.rp 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#
```

要发现 AL2023 更新，请执行以下一项或多项操作：

- 运行 `dnf check-update` 命令。这将检查您锁定的 Amazon Linux 版本中任何未应用的更新。如果您仅更新了 `system-release` 程序包，改变了实例锁定的存储库版本但未应用其中的任何可用更新，则仍可能会显示更新。
- 订阅 Amazon Linux 存储库更新 SNS 主题 (`arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates`)。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的订阅 Amazon SNS 主题。
- 定期参阅AL2023发行说明。
- 通过 使用 dnf check-release-update 检查较新的存储库版本 发现新版本。

> **Important**
>
> AL2023 包含安全更新的新版本经常发布。请务必及时了解相关的安全补丁。

（安全）更新后自动重启服务

Amazon Linux 现在随附 smart-restart 程序包。Smart-restart 在使用系统程序包管理器安装或删除程序包时，会在系统更新后重启 systemd 服务。这在每次执行 dnf （update|upgrade|downgrade）时都会发生。

Smart-restart 使用来自 dnf-utils 的 needs-restarting 程序包和自定义拒绝列表机制来确定需要重启哪些服务以及是否建议系统重启。如果建议系统重启，则会生成一个重启提示标记文件（/run/smart-restart/reboot-hint-marker）。

要安装 **smart-restart**，请执行以下操作

运行以下 DNF 命令（就像处理任何其他程序包一样）。

```
$ sudo dnf install smart-restart
```

安装后，后续的事务将触发 smart-restart 逻辑。

拒绝列表

可以指示 Smart-restart 阻止重启某些服务。被阻止的服务将不会影响是否需要重启的决定。要阻止其他服务，请按以下示例所示，在 /etc/smart-restart-conf.d/ 中添加一个后缀为 -denylist 的文件。

```
$ cat /etc/smart-restart-conf.d/custom-denylist
# Some comments
myservice.service
```

> **Note**
>
> 在决定是否需要重启时，所有 *-denylist 文件都会被读取和评估。

自定义钩子

除了拒绝列表，smart-restart 还提供了一种机制，在尝试重启服务之前和之后运行自定义脚本。自定义脚本可用于手动执行准备步骤，或通知其他组件重启是否剩余或已完成。

所有在 /etc/smart-restart-conf.d/ 中带有后缀 -pre-restart 或 -post-restart 的脚本都会被执行。如果顺序重要，请为所有脚本添加数字前缀以确保执行顺序，如下例所示。

```
$ ls /etc/smart-restart-conf.d/*-pre-restart
001-my-script-pre-restart
002-some-other-script-pre-restart
```

何时需要重启以应用安全更新？

在某些情况下，Amazon Linux 需要重启以应用更新：

- Linux 内核程序包的更新需要重启以激活带有最新安全更新的新内核。内核实时修补可能允许您在有限时间内推迟安全更新。有关详细信息，请参阅 AL2023 上的内核实时修补。
- 在 EC2 Metal 实例上，Amazon Linux 提供微码更新（通过英特尔的`microcode_ctl`软件`amd-ucode-firmware`包 CPUs 和 AMD CPU 的软件包）。这些微码更新只会在后续实例重启时激活。对于虚拟化 EC2 实例，底层的 AWS Nitro 系统会为您处理微码更新。
- 一些正在运行的 systemd 服务只有在系统完全重启后才能正常工作。`smart-restart` 机制将通过显示重启提示来通知您此类情况。请参阅（安全）更新后自动重启服务。

启动已启用最新存储库版本的实例

您可以在用户数据脚本中添加 DNF 命令，以控制当 Amazon Linux AMI 启动时，其上可以安装哪些 RPM 软件包。在以下示例中，使用用户数据脚本来确保通过其启动的任何实例都安装了同样的软件包更新。

```
#!/bin/bash
dnf upgrade --releasever=2023.0.20230210
# Additional setup and install commands below
dnf install httpd php7.4 mysql80
```

您必须以超级用户（根用户）的身份来运行此脚本。为此，请运行以下命令。

```
$ sudo sh -c "bash nameofscript.sh"
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的用户数据和 shell 脚本。

> **Note**
>
> 您也可以不使用用户数据脚本，而是启动最新的 Amazon Linux AMI 或基于 Amazon Linux AMI 的自定义 AMI。最新的 Amazon Linux AMI 已安装了所有必要的更新，并且被配置为指向特定的存储库版本。

获取程序包支持信息

AL2023 包含许多不同的开源软件项目。这些项目中的每一个都独立于 Amazon Linux 进行管理，并且有不同的发布和 end-of-support时间表。为了向您提供有关这些不同软件包的 Amazon Linux 特定信息，DNF `supportinfo` 插件提供了软件包的元数据。在以下示例中，**`dnf supportinfo`** 命令返回 `glibc` 软件包的元数据。

```
$ 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 支持信息也可以在AL2023 发行说明的支持声明部分中找到。

使用 **`dnf check-release-update`** 检查较新的存储库版本

在 AL2023 实例中，您可以使用该DNF实用程序来管理存储库和应用更新的RPM软件包。可以从 Amazon Linux 存储库获得这些软件包。您可以使用 DNF 命令 `dnf check-release-update` 来查看 DNF 存储库的新版本。

> **Note**
>
> AL2023 默认情况下，容器镜像不包含该`dnf check-release-update`命令。

```
$ 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)'"
```

当运行 dnf install 'dnf-command(check-release-update)' 时，dnf 将安装提供 check-release-update 命令的程序包，即 dnf-plugin-release-notification 程序包。在以下示例中，向 dnf 提供了 -q 参数以使其具有静默输出。

```
$ dnf -y -q install 'dnf-command(check-release-update)'
Installed:
  dnf-plugin-release-notification-1.2-1.amzn2023.0.2.noarch
```

在非容器化环境（例如 Amazon EC2 实例）中，默认包含 check-release-update 命令。

```
$ 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
```

这将返回所有可用的 DNF 存储库的更新版本的完整列表。如果未返回任何内容，则说明 DNF 当前已配置为使用最新的可用版本。当前安装的 system-release 软件包版本设置 releasever DNF 变量。要查看当前存储库版本，请运行以下命令。

```
$ rpm -q system-release --qf "%{VERSION}\n"
```

当您运行 DNF 软件包事务（例如安装、更新或删除命令）时，会出现一条警告消息，告知您有任何新的存储库版本。例如，如果您将httpd软件包安装在从旧版本启动的实例上 AL2023，则会返回以下输出。

```
$ 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!
```

添加、启用或禁用新存储库

> **Warning**
>
> 仅添加设计用于的存储库 AL2023。
> 虽然为其他发行版设计的存储库现在可以使用，但不能保证在任何软件包更新 AL2023 或存储库不是为与之配合使用而设计的，它们会继续运行 AL2023。

要从默认 Amazon Linux 存储库以外的其他存储库安装程序包，您需要配置 DNF 程序包管理系统以告知其存储库的位置。

要告知 dnf 有关程序包存储库的信息，请将存储库信息添加到 /etc/yum.repos.d/ 目录中该存储库的配置文件中。许多第三方存储库要么提供配置文件内容，要么提供一个包含配置文件的可安装的程序包。

> **Note**
>
> 虽然可以直接在 /etc/dnf/dnf.conf 文件中配置存储库，但不建议这样做。建议每个存储库在 /etc/yum.repos.d/ 目录下的各自文件中进行配置。

要了解当前启用了哪些存储库，可以运行以下命令：

```
$ 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**
>
> 如果不添加 --verbose 选项标志，则输出仅包含 Repo-id、Repo-name 和 Repo-status 信息。

将 **yum** 存储库添加到 **/etc/yum.repos.d** 目录

1. 查找 .repo 文件的位置。在本示例中，.repo 文件位于 *https://www.example.com/repository.repo*。
2. 使用 dnf config-manager 命令添加存储库。

```
$ 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
```

安装存储库后，必须按照以下过程启用存储库。

要在中启用yum存储库/etc/yum.repos.d，请使用带有--enable标志和*repository*名称的dnf config-manager命令。

```
$ sudo dnf config-manager --enable repository
```

> **Note**
>
> 要禁用存储库，请使用同样的命令语法，但在命令中将 --enable 替换为 --disable。

使用 cloud-init 添加存储库

除了使用前面的方法添加存储库，您还可以使用 cloud-init 框架添加新的存储库。

要添加新的软件包存储库，建议您使用以下模板。考虑将此文件保存在本地。

```
#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**
>
> 使用 cloud-init 的一个好处是，您可以在配置文件中添加一个 packages: 部分。在该部分中，您可以包括要安装的软件包的名称。您可以安装来自默认存储库或来自您在 cloud-config 文件中添加的新存储库的软件包。
> 有关 YAML 文件结构的更多具体信息，请参阅《cloud-init 文档》中的添加 YUM 存储库。

设置 YAML 格式文件后，可以在 AWS CLI中在 cloud-init 框架中运行该文件。确保包含 --userdata 选项和 .yml 文件的名称，以便调用所需操作。

```
$ 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
```

AL2023 上的内核实时修补

您可以使用 AL2023 的内核实时修补功能，将特定的安全漏洞和严重错误补丁应用到正在运行的 Linux 内核，而无需重启或中断运行的应用程序。此外，在内核实时修补应用这些修复期间，直到系统可以重启之前，它有助于提高应用程序的可用性。

AWS 为 AL2023 发布两种类型的内核实时补丁：

- 安全更新 - 包括 Linux 常见漏洞和风险（CVE）的更新。通常使用 Amazon Linux 安全通告评级将这些更新评为重要 或关键。它们通常对应于通用漏洞评分系统 (CVSS) 的 7 分或更高。在某些情况下，AWS 可能会在分配 CVE 之前提供更新。在这些情况下，补丁可能会显示为错误修复。
- 错误修复 - 包括与 CVE 无关的关键错误和稳定性问题的修复。

AWS 为 AL2023 内核版本提供长达 3 个月的内核实时补丁支持。三个月之后，您必须更新为更高的内核版本才能继续收到内核实时补丁。

AL2023 内核实时补丁以现有 AL2023 存储库中签名的 RPM 软件包形式提供给客户使用。可以使用现有 DNF 软件包管理器工作流在单个实例上安装补丁。或者，使用 AWS Systems Manager 在一组托管实例上安装补丁。

AL2023 上的内核实时修补是免费提供的。

主题

限制

在应用内核实时补丁时，无法执行休眠以及使用高级调试工具（例如 `SystemTap`、`kprobes` 和基于 eBPF 的工具），或者访问内核实时修补基础设施使用的 `ftrace` 输出文件。

> **Note**
>
> 由于技术限制，有些问题无法通过实时修补解决。因此，这些修复将不会在内核实时修补程序包中提供，而仅在本机内核程序包更新中提供。您可以照常安装本机内核程序包并更新和重启系统以激活补丁。

支持的配置和先决条件

运行 AL2023 的 Amazon EC2 实例和本地虚拟机支持内核实时修补。

要使用 AL2023 上的内核实时修补，必须满足以下先决条件：

- 64 位 x86_64 或 ARM64 架构
- 内核版本 6.1 或 6.12

策略要求

要从 AL2023 存储库下载程序包，Amazon EC2 需要访问服务拥有的 Amazon S3 存储桶。如果您在环境中使用 Amazon S3 的 Amazon 虚拟私有云（VPC）端点，请确保您的 VPC 端点策略允许访问那些公共存储桶。下表描述了 Amazon EC2 可能需要访问以用于内核实时修补的 Amazon S3 存储桶。

S3 存储桶 ARN	描述
arn:aws:s3:::al2023-repos-*region*-de612dc2/*	包含 AL2023 存储库的 Amazon S3 存储桶。

使用内核实时修补

要在单独的实例上启用和使用内核实时修补，可在该实例本身上使用命令行。要在一组托管实例上启用和使用内核实时修补，可以使用 AWS Systems Manager。

以下部分介绍如何在单独的实例上使用命令行来启用和使用内核实时修补。

有关在一组托管实例上启用和使用内核实时修补的更多信息，请参阅《AWS Systems Manager 用户指南》中的在 AL2023 实例上使用内核实时修补。

主题

启用内核实时修补

默认情况下，AL2023 上的内核实时修补处于禁用状态。要使用内核实时修补，必须为其安装 DNF 插件，并启用实时修补功能。

启用内核实时修补

1. 内核实时补丁适用于具有内核版本 6.1 的 AL2023。要检查内核版本，请运行以下命令。

```
$ sudo dnf list kernel
```

2. 为内核实时修补安装 DNF 插件。

```
$ sudo dnf install -y kpatch-dnf
```

3. 为内核实时修补启用 DNF 插件。

```
$ sudo dnf kernel-livepatch -y auto
```

此命令还会安装来自配置的存储库的最新版本的内核实时补丁 RPM。

4. 要确认内核实时修补的 DNF 插件是否安装成功，请运行以下命令。

当您启用内核实时修补时，会自动应用空的内核实时补丁 RPM。如果内核实时修补成功启用，此命令返回一个列表，其中包括初始的空内核实时修补 RPM（以及另一个设置包含实时补丁的 DNF 存储库的 RPM）。

```
$ 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. 安装 kpatch 软件包。

```
$ sudo dnf install -y kpatch-runtime
```

6. 如果之前安装过 kpatch 服务，请更新它。

```
$ sudo dnf upgrade kpatch-runtime
```

7. 启动 kpatch 服务。此服务在初始化或启动时会加载所有内核实时补丁。

```
$ sudo systemctl enable kpatch.service && sudo systemctl start kpatch.service
```

查看可用的内核实时补丁

Amazon Linux 安全警报会发布到 Amazon Linux 安全中心。有关 AL2023 安全警报（包括内核实时补丁的警报）的更多信息，请参阅 Amazon Linux 安全中心。内核实时补丁的前缀为 ALASLIVEPATCH。Amazon Linux 安全中心可能不会列出解决错误的内核实时补丁。

您还可以使用命令行搜索通告和 CVE 的可用内核实时补丁。

列出所有可用的内核实时补丁以获取通告

使用以下命令。

```
$ 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
```

列出 CVE 的所有可用内核实时补丁

使用以下命令。

```
$ 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
```

应用内核实时补丁

您可以使用 DNF 软件包管理器，就像应用常规更新那样来应用内核实时补丁。用于内核实时修补的 DNF 插件管理可用的内核实时补丁。

> **Tip**
>
> 我们建议您定期使用内核实时修补更新内核，以确保在系统可以重启之前接收特定的重要和关键安全修复。还请检查是否有其他修复已可用于本机内核程序包，这些修复无法作为实时补丁部署，并在这些情况下更新并重启到内核更新。

您可以选择应用特定的内核实时补丁，或者应用任何可用的内核实时补丁以及定期安全更新。

应用特定内核实时补丁

1. 使用 查看可用的内核实时补丁 中描述的命令之一获取内核实时补丁版本。
2. 为您的 AL2023 内核应用内核实时补丁。

```
$ sudo dnf install kernel-livepatch-kernel_version-package_version.amzn2023.x86_64
```

例如，以下命令为 AL2023 内核版本 6.1.12-17.42 应用内核实时补丁。

```
$ sudo dnf install kernel-livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
```

应用任何可用的内核实时补丁以及定期安全更新

使用以下命令。

```
$ sudo dnf upgrade --security
```

省略 `--security` 选项将包含错误修复。

Important

- 应用内核实时补丁后，内核版本不会更新。仅当重启实例后，版本才会更新为新版本。
- 一个 AL2023 内核可收到为期三个月的内核实时补丁。之后，不会为该内核版本发布新的内核实时补丁。
- 要想在三个月后继续收到内核实时补丁，您必须重启实例以移至新的内核版本。更新后，实例将在接下来的三个月内继续收到内核实时补丁。
- 要查看内核版本的支持窗口，请运行以下命令。

```
$ 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.
```

查看应用的内核实时补丁

查看应用的内核实时补丁

使用以下命令。

```
$ 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)
```

该命令返回已加载和安装的安全更新内核实时补丁的列表。下面是示例输出。

> **Note**
>
> 单个内核实时补丁可以包含和安装多个实时补丁。

禁用内核实时修补

如果您不再需要使用内核实时修补，可以随时禁用它。

- 要禁用 livepatches，可执行以下操作：
  1. 禁用插件：

     ```
     $ sudo dnf kernel-livepatch manual
     ```

  2. 禁用 kpatch 服务：

     ```
     $ sudo systemctl disable --now kpatch.service
     ```

- 要完全移除 livepatch 工具，可执行以下操作：
  1. 移除插件：

     ```
     $ sudo dnf remove kpatch-dnf
     ```

  2. 移除 kpatch-runtime：

     ```
     $ sudo dnf remove kpatch-runtime
     ```

  3. 移除任何已安装的 livepatches：

```
$ sudo dnf remove kernel-livepatch\*
```

正在更新 Linux 内核 AL2023

主题

Linux 内核版本开启 AL2023

AL2023 定期包括基于 Linux 内核的长期支持 (LTS) 版本的新内核版本。

AL2023 最初于 2023 年 3 月发布，内核为 6.1。

2025 年 4 月， AL2023 增加了对 Linux 内核 6.12 的支持。该内核增加了新功能，包括 EEVDF 调度、FUSE 直通 I/O 支持、新的 Futex API 以及 eBPF 的改进。内核 6.12 还允许用户空间程序在运行时使用用户空间影子栈和内存密封来保护自身。

2026 年 3 月， AL2023 增加了对 Linux 内核 6.18 的支持。更新后的内核 6.18 在处理器支持、虚拟化、安全性和性能方面带来了更多改进。值得注意的功能包括改进的跨架构的 IOMMU 功能以及用于管理 CPU 漏洞缓解措施的攻击向量控制。性能增强来自密码学优化、更快的 FSCRYPT 操作、内存管理改进以及引入 Sheaves 作为新的可选的、基于每个 CPU 阵列的缓存层。

更新 AL2023 到较新的内核版本

从 2026 年 6 月开始， AL2023 将每年更新默认内核。这`al2023-ami-kernel-default`组 AMIs 将更新到最新的 LTS 内核，因此新启动的实例将自动提供新的内核版本——这是了解最新安全修复和性能改进的最简单方法。

如果您更喜欢选择特定的内核版本，则可以通过选择预装了所需内核的 AMI 或升级现有的 AL2023 EC2 实例来运行带有内核 6.12 或 6.18 的 AL2023。AL2023 AL2023

运行具有特定内核版本的 AL2023 AMI

您可以选择通过 AWS 控制台或查询 SSM 以获取特定参数来运行预装了特定内核的 AL2023 AMI。要查询的 SSM 密钥以 `/aws/service/ami-amazon-linux-latest/` 开头，后跟以下之一

适用于内核 6.12

- `al2023-ami-kernel-6.12-arm64`，适用于 arm64 架构
- `al2023-ami-minimal-kernel-6.12-arm64`，适用于 arm64 架构（AMI 最低版本）
- `al2023-ami-kernel-6.12-x86_64`，适用于 x86_64 架构
- `al2023-ami-minimal-kernel-6.12-x86_64`，适用于 x86_64 架构（AMI 最低版本）

适用于内核 6.18

- `al2023-ami-kernel-6.18-arm64`，适用于 arm64 架构
- `al2023-ami-minimal-kernel-6.18-arm64`，适用于 arm64 架构（AMI 最低版本）
- `al2023-ami-kernel-6.18-x86_64`，适用于 x86_64 架构
- `al2023-ami-minimal-kernel-6.18-x86_64`，适用于 x86_64 架构（AMI 最低版本）

有关选择[AL2023 使用 SSM 参数启动和 AWS CLI]的详细信息，请参阅 AL2023 AMIs。

将 AL2023 实例更新到更新的内核

您可以通过以下步骤将正在运行的 AL2023 实例就地升级到内核 6.12 或 6.18：

1. 检测当前内核并设置目标版本：

```
# Automatically detect current kernel version BEFORE upgrade
$ CURRENT_KERNEL=$(uname -r)
$ SOURCE_VERSION=""

$ if [[ $CURRENT_KERNEL == *"6.12"* ]]; then
    SOURCE_VERSION="6.12"
else
    SOURCE_VERSION=""
fi

# Save the source version to a persistent location for use after reboot
$ echo "${SOURCE_VERSION}" | sudo tee /var/lib/source_kernel_version > /dev/null
```

```
# Set your target version (change this to your desired kernel: 6.12 or 6.18)
$ TARGET_VERSION="6.12"
```

```
$ echo "Current kernel: ${SOURCE_VERSION:-6.1}"
$ echo "Upgrading to kernel ${TARGET_VERSION}"
```

2. 安装目标内核软件包：

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

3. 获取目标内核包的最新版本：

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |
 sort -V | tail -1)
```

4. 将新内核设为默认内核：

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

5. 重启系统：

```
$ sudo reboot
```

6. 卸载之前的内核：

```
# Read the source kernel version from the saved file
$ SOURCE_VERSION=$(sudo cat /var/lib/source_kernel_version)

# Uninstall the source kernel
$ sudo dnf remove -y kernel${SOURCE_VERSION}
```

7. 将多余的内核包替换为目标内核等效包：

```
# Set your target version (change this to your desired kernel: 6.12 or 6.18)
$ TARGET_VERSION="6.12"
```

```
$ declare -A pkgs
$ pkgs=(
[bpftool${SOURCE_VERSION}]=bpftool${TARGET_VERSION}
[kernel${SOURCE_VERSION}-debuginfo]=kernel${TARGET_VERSION}-debuginfo
[kernel${SOURCE_VERSION}-debuginfo-common]=kernel${TARGET_VERSION}-debuginfo-common
[kernel${SOURCE_VERSION}-headers]=kernel${TARGET_VERSION}-headers
[kernel${SOURCE_VERSION}-libbpf]=kernel${TARGET_VERSION}-libbpf
[kernel${SOURCE_VERSION}-libbpf-devel]=kernel${TARGET_VERSION}-libbpf-devel
```

```
[kernel${SOURCE_VERSION}-libbpf-static]=kernel${TARGET_VERSION}-libbpf-static
[kernel${SOURCE_VERSION}-modules-extra-common]=kernel${TARGET_VERSION}-modules-
extra-common
[kernel${SOURCE_VERSION}-tools]=kernel${TARGET_VERSION}-tools
[kernel${SOURCE_VERSION}-tools-devel]=kernel${TARGET_VERSION}-tools-devel
[perf${SOURCE_VERSION}]=perf${TARGET_VERSION}
[python3-perf${SOURCE_VERSION}]=python3-perf${TARGET_VERSION}
)
$ for pkg in "${!pkgs[@]}"; do
  rpm -q $pkg && sudo dnf -y swap $pkg "${pkgs["$pkg"]}" ;
done
```

8. （可选）卸载先前内核版本的 kernel-devel：

```
$ rpm -q kernel${SOURCE_VERSION}-devel && sudo dnf remove -y kernel
${SOURCE_VERSION}-devel
```

降级到较早的内核版本

如果您在任何时候需要降级回较早的内核版本，请使用以下步骤：

1. 检测当前内核并设置目标版本：

```
# Automatically detect current kernel version BEFORE downgrade
$ CURRENT_KERNEL=$(uname -r)
$ SOURCE_VERSION=""

$ if [[ $CURRENT_KERNEL == *"6.12"* ]]; then
    SOURCE_VERSION="6.12"
elif [[ $CURRENT_KERNEL == *"6.18"* ]]; then
    SOURCE_VERSION="6.18"
fi

# Save the source version to a persistent location for use after reboot
$ echo "${SOURCE_VERSION}" | sudo tee /var/lib/source_kernel_version > /dev/null
```

```
# Set your target version (change this to your desired kernel)
# Use "" for kernel 6.1, "6.12" for kernel 6.12
$ TARGET_VERSION=""
```

```
$ echo "Downgrading from kernel ${SOURCE_VERSION:-6.1} to kernel
 ${TARGET_VERSION:-6.1}"
```

2. 将多余的内核包替换为目标内核等效包：

```
$ declare -A pkgs
$ pkgs=(
[bpftool${TARGET_VERSION}]=bpftool${SOURCE_VERSION}
[kernel${TARGET_VERSION}-debuginfo]=kernel${SOURCE_VERSION}-debuginfo
[kernel${TARGET_VERSION}-debuginfo-common]=kernel${SOURCE_VERSION}-debuginfo-common
[kernel${TARGET_VERSION}-headers]=kernel${SOURCE_VERSION}-headers
[kernel${TARGET_VERSION}-libbpf]=kernel${SOURCE_VERSION}-libbpf
[kernel${TARGET_VERSION}-libbpf-devel]=kernel${SOURCE_VERSION}-libbpf-devel
[kernel${TARGET_VERSION}-libbpf-static]=kernel${SOURCE_VERSION}-libbpf-static
[kernel${TARGET_VERSION}-modules-extra-common]=kernel${SOURCE_VERSION}-modules-
extra-common
[kernel${TARGET_VERSION}-tools]=kernel${SOURCE_VERSION}-tools
[kernel${TARGET_VERSION}-tools-devel]=kernel${SOURCE_VERSION}-tools-devel
[perf${TARGET_VERSION}]=perf${SOURCE_VERSION}
[python3-perf${TARGET_VERSION}]=python3-perf${SOURCE_VERSION}
)
$ for pkg in "${!pkgs[@]}"; do
  rpm -q "${pkgs["$pkg"]}" && sudo dnf -y swap "${pkgs["$pkg"]}" $pkg ;
done
```

3. 安装目标内核软件包：

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

4. 获取目标内核包的最新版本：

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |
 sort -V | tail -1)
```

5. 将目标内核设为默认内核：

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

6. 重启系统：

```
$ sudo reboot
```

7. 卸载源内核：

```
# Read the source kernel version from the saved file
$ SOURCE_VERSION=$(sudo cat /var/lib/source_kernel_version)

# Uninstall the source kernel
$ sudo dnf remove -y kernel${SOURCE_VERSION}
```

AL2023 内核-常见问题解答

1. 内核更新后是否需要重启？

对运行中内核的任何更改都需要重启。

2. 如何 up-to-date跨多个实例保留内核？

Amazon Linux 不提供管理实例集的工具。我们建议您使用诸如 AWS Systems Manager 等工具为大型实例集打补丁。

3. 如何检查当前运行的内核版本？

在您的 AL2023 实例上执行以下命令：

```
$ uname -r
```

4. AL2023 推荐我使用哪个内核？

建议在所有其他 AL2023 内核仍受支持的情况下升级到最新的 AL2023 内核 6.18。建议客户在升级之前对其工作负载进行测试。

5. 我现有的应用程序可以与任何 AL2023 内核一起使用吗？

AL2023 支持更新的内核（6.12 或 6.18），就像内核 6.1 一样。应用程序将起作用，并且正在幕后进行改进。无论如何，在切换到更新的内核之前，客户都应测试其特定工作负载。

6. 如何为内核 6.12 或 6.18 安装内核头文件、开发包和额外模块？

请运行：

```
$ version=$(uname -r | grep -oP '^\d+\.\d+')
$ sudo dnf install -y kernel${version}-modules-extra-$(uname -r) kernel${version}-
headers-$(uname -r) kernel${version}-devel-$(uname -r)
```

7. 内核 6.12 和 6.18 将支持多长时间？

在亚马逊 Linux 2023 的计划生命周期结束之前，也就是 2029-06-30，将一直支持内核 6.12 和 6.18。

开启运行时编程入门 AL2023

AL2023 提供了某些语言运行时的不同版本。我们与同时支持多个版本的上游项目合作。查找有关如何使用 dnf 命令搜索、安装和管理这些名称受版本控制的软件包的信息。

以下主题概述了每种语言生态系统的存在方式 AL2023。

主题

AL2023 中的 C、C++ 和 Fortran

AL2023 包含 GNU 编译器集合 (GCC) 以及用于 LLVM（低级虚拟机）的 Clang 前端。

GCC 的主要版本在 AL2023 的整个生命周期中将保持不变。其次要版本带有错误修复，可能包含在 AL2023 版本中。其他错误、性能和安全修复可能会向后移植到 AL2023 附带的 GCC 的主要版本中。

AL2023 包含版本 11 的 GCC 作为默认编译器，带有 C (`gcc`)、C++ (`g++`) 和 Fortran (`gfortran`) 前端。此外，AL2023 提供版本 14 的 GCC 作为可选的替代编译器，可与默认版本一起安装。

AL2023 未启用 Ada (`gnat`)、Go (`gcc-go`)、Objective-C 或 Objective-C++ 前端。

构建 AL2023 RPM 时使用的默认编译器标志包括优化和强化标志。要使用 GCC 构建您自己的代码，我们建议您包含优化和强化标志。

> **Note**
>
> 调用 `gcc --version` 时，会显示一个版本字符串，例如 `gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4)`。`Red Hat` 指的是 Amazon Linux GCC 软件包所基于的 GCC 供应商分支。根据 `gcc --help` 显示的错误报告 URL，所有错误报告和支持请求均应指向 Amazon Linux。
> 有关此供应商分支中一些长期变更（例如 `__GNUC_RH_RELEASE__` 宏）的更多信息，请参阅 Fedora 程序包源码。

有关核心工具链的更多信息，请参阅 Core 工具链软件包 glibc、gcc、binutils。

有关 AL2023 及其与其他 Linux 发行版关系的更多信息，请参阅 与 Fedora 的关系。

有关 AL2023 与 AL2 相比编译器三元组变更的更多信息，请参阅 编译器三元组。

主题

- GCC 14
- 语言标准版本比较

GCC 14

AL2023 提供 GCC 14 作为可选编译器，可与默认的 GCC 11 一起安装。GCC 14 包含最新的语言特性和优化，适用于需要较新 C、C++ 或 Fortran 标准支持的项目。

要安装 GCC 14，请使用以下命令：

```
sudo dnf install gcc14 gcc14-c++ gcc14-gfortran
```

GCC 14 编译器安装时使用特定于版本的命令名称，以避免与默认的 GCC 11 冲突：

- `gcc14-gcc`：C 编译器
- `gcc14-g++`：C++ 编译器
- `gcc14-gfortran`：Fortran 编译器

示例用法：

```
gcc14-gcc -o myprogram myprogram.c
```

```
gcc14-g++ -o mycppprogram mycppprogram.cpp
gcc14-gfortran -o myfortranprogram myfortranprogram.f90
```

您可以通过运行以下命令验证安装的版本：

```
gcc14-gcc --version
```

这将显示类似以下的版本信息：`gcc14-gcc (GCC) 14.2.1 20250110 (Red Hat 14.2.1-7)`

> **Note**
>
> GCC 11 和 GCC 14 可以同时安装在同一系统上。默认的 `gcc`、`g++` 和 `gfortran` 命令将继续使用 GCC 11，而 GCC 14 通过特定于版本的命令访问。

语言标准版本比较

下表比较了不同 Amazon Linux 版本和 GCC 编译器版本的默认语言标准版本：

Amazon Linux 版本	C 标准（默认）	C++ 标准（默认）	Fortran 标准
AL2 附带 GCC 7（默认）	C11（201112L）	C++14（201402L）	Fortran 2008
AL2 附带 GCC 10（可选）	C17/C18（201710L）	C++14（201402L）	Fortran 2008
AL2023 附带 GCC 11（默认）	C17/C18（201710L）	C++17（201703L）	Fortran 2008
AL2023 附带 GCC 14（可选）	C17/C18（201710L）	C++17（201703L）	Fortran 2008

各 GCC 版本的主要改进：

- GCC 10 对比 GCC 7：将默认 C 标准从 C11 升级到 C17/C18，增加了对 C++20 特性的支持，并改进了优化能力。

- GCC 11 对比 GCC 10：将默认 C++ 标准从 C++14 升级到 C++17，增强了对 C++20 的支持，并增加了实验性的 C++23 特性。
- GCC 14 对比 GCC 11：增加了完整的 C23 标准支持，增强了 C++23 特性，改进了优化，并提高了标准符合性。

支持的语言标准：

- C 标准：所有版本均支持 C90、C99、C11 和 C17/C18。GCC 10+ 支持 C2x（C23 草案），而 GCC 14 提供完整的 C23 支持。
- C++ 标准：所有版本均支持 C++98、C++03、C++11、C++14、C++17 和 C++20。GCC 11+ 提供实验性的 C++23 支持，GCC 14 则提供增强的 C++23 特性。
- Fortran 标准：所有版本主要支持 Fortran 2008，并根据 GCC 版本提供不同程度的 Fortran 2018 特性支持。

> **Note**
>
> 虽然 GCC 11 和 14 的默认标准保持一致，但 GCC 14 在显式使用 `-std=` 标志请求时，提供了显著改进的语言特性支持、更好的优化、增强的诊断信息以及对新标准更完整的实现。

Go在 AL2023

您可能想在 Amazon Linux Go上构建自己的代码，也可能需要使用随附 AL2023的工具链。类似于 AL2， AL2023 将在操作系统的整个生命周期中更新Go工具链。这或许是为了回应我们发布的工具链中的任何 CVE，或许是季度发布的一部分。

Go 是一门发展相对迅速的语言。可能会出现用 Go 编写的现有应用程序必须适应新版本 Go 工具链的情况。有关 Go 的更多信息，请参阅 Go 1 与 Go 程序的未来。

尽管 AL2023 将在Go工具链的生命周期中加入新版本，但这不会与上游Go版本保持一致。因此，如果您想使用Go语言和标准库的尖端功能来构建Go代码，则 AL2023 可能不适合使用中提供的Go工具链。

在的生命周期内 AL2023，不会从存储库中删除以前的软件包版本。如果需要先前的 Go 工具链，您可以选择放弃较新 Go 工具链的错误和安全修复，并使用适用于任何 RPM 的相同机制从存储库安装旧版本。

如果你想在上面构建自己的Go代码 AL2023，你可以使用中包含的Go工具链，因为你知道这个工具链可能会在 AL2023 整个生命周期中向前发展。 AL2023

AL2023 写入的 Lambda 函数 Go

由于 Go 编译为原生代码，Lambda 将 Go 视为自定义运行时。您可以使用`provided.al2023`运行时将Go函数部署 AL2023 到 Lambda 上。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的使用 Go 构建 Lambda 函数。

AL2023 中的 Java

AL2023 提供多个版本的 Amazon Corretto 以支持基于 Java 的工作负载。AL2023 中包含的所有基于 Java 的程序包均使用 Amazon Corretto 17 构建。

Corretto 是一个由 Amazon 提供长期支持的 Open Java Development Kit（OpenJDK）构建版本。Corretto 使用 Java 技术兼容性工具包（TCK）进行认证，以确保其符合 Java SE 标准，并在 Linux、Windows 和 macOS 上可用。

Corretto 1.8.0、Corretto 11 和 Corretto 17 都有一个可用的 Amazon Corretto 软件包。

AL2023 中的每个 Corretto 版本受支持的时间与 Corretto 版本相同，或者直到 AL2023 的使用周期结束，以较早者为准。更多信息，请参阅 Amazon Linux 程序包支持声明和 Amazon Corretto 常见问题解答。

Node.js在 AL2 023

Node.js在 AL2 023 中，由 20、22 和 24 版本表示。Amazon Linux 遵循上游支持计划，可以随时在 Package 支持状态页面上查看任何 Node.js 版本的支持状态。所有受支持的 Node.js 版本都是命名空间化的，并且可以同时安装在同一系统上。命名空间化确保每个 Node.js 安装在文件系统内是唯一的。这是通过根据运行时版本重命名关键目录和文件来实现的。实际的可执行文件名称将类似于 node-{MAJOR_VERSION} 或 npm-{MAJOR_VERSION}。但是，一次只能有一个 Node.js 版本处于活动状态。此活动版本提供了默认目录和文件名，例如 n ode、npm 或/usr/lib/node_mod ule s，将它们指向当前活动的运行时。

这是利用 alternatives 工具的功能实现的。重要的是要记住，默认的可执行文件名是虚拟的，并且在指向不同安装的 Node.js 版本时可以随时更改。这种灵活性使得使用 node 在 shebang 中的软件可以在被调用时选择所需的版本。但是，当需要特定版本的Node.js时，可以通过调用命名空间可执行文件

（例如 node -20 或 nod e-22）来实现版本的持久性，该可执行文件将始终使用指定的运行时版本。此外，npm 工具的命名空间可执行文件，如 npm-20 或 npm-22，总是与相应的 Node.js 版本相关联，无论当前活动的运行时是什么。

Node.js以几个以 “” `nodejs{MAJOR_VERSION}` 开头的命名空间包的形式分发。这些软件包提供 n ode、npm 工具的兼容版本、文档、库等。例如，Node.js22 中的节点和 npm 分别由`nodejs22`和`nodejs22-npm`包提供。

alternatives 工具提供了一个用于在 Node.js 版本之间切换的单一命令。默认情况下，alternatives 被配置为处于自动模式，该模式使用优先级来确定当前活动的 Node.js 版本。然而，您可以随时激活任何已安装的版本。目前，所有受支持的 Node.js 版本具有相等的优先级，这意味着第一个安装的版本将自动激活。

使用 alternatives 的一些有用示例

1. 检查 alternatives 的配置内容

```
alternatives --list
```

2. 检查 node 的当前配置

```
alternatives --display node
```

3. 交互式更改 Node.js 版本

```
alternatives --config node
```

4. 切换到手动模式并选择特定版本

```
alternatives --set node /usr/bin/node-{MAJOR_VERSION}
```

5. 切换回自动版本选择模式

```
alternatives --auto node
```

AL2023 中的 Perl

AL2023 提供了 Perl 编程语言的 5.32 版本。

尽管 Perl 在过去几十年中作为 Perl 5 版本的一部分提供了高度的语言兼容性，但 Amazon Linux 在 AL2023 发布期间预计不会从 Perl 5.32 迁移。Amazon Linux 将在 AL2023 的整个生命周期内根据我们的程序包支持声明继续为 Perl 提供安全补丁。

AL2023 中的 Perl 模块

各种 Perl 模块在 AL2023 中被打包为 RPM。尽管有许多 Perl 模块可作为 RPM 使用，但 Amazon Linux 并不旨在打包所有可能的 Perl 模块。打包为 RPM 的模块可能被其他操作系统 RPM 程序包所依赖，因此 Amazon Linux 将优先处理这些安全补丁，而不是纯粹的功能更新。

AL2023 还包括 CPAN，以便 Perl 开发人员可以使用惯用的程序包管理器来管理 Perl 模块。

PHP在 AL2 023

AL2023 目前提供PHP编程语言，版本 8.1、8.2、8.3、8.4 和 8.5。每个版本的支持时间与上游 PHP 相同。有关更多信息，请参阅程序包支持声明。

从旧的 PHP 版本迁移

上游 PHP 社区整理了用于迁移的全面文档：

- 从 PHP 8.4.x 到 8.5.x PHP
- 从 PHP 8.3.x 迁移至 PHP 8.4.x
- 从 PHP 8.2.x 迁移至 PHP 8.3.x
- 从 PHP 8.1.x 迁移至 PHP 8.2.x
- 从 PHP 8.0.x 迁移至 PHP 8.1.x

AL2 包括 PHP 8.0、8.1 和 8.2，`amazon-linux-extras`以便轻松升级到 AL2 023。

从 PHP 7.x 版本迁移

> **Note**
>
> PHP 项目维护着受支持版本的列表和时间表，以及不受支持分支的列表。
> AL2023 发布时，PHP社区不支持所有 7.x 和 5.x 版本，PHP也未作为选项包含在 023 中。
> AL2

上游 PHP 社区整理了从 PHP 7.4 迁移至 PHP 8.0 的全面文档。结合上一节中关于迁移到 PHP 8.1 和 PHP 8.2 所引用的文档，您可以将基于 PHP 的应用程序迁移到现代 PHP。

> **Note**
>
> AL2 包括 PHP 7.1、7.2、7.3 和 7.4 英寸`amazon-linux-extras`。值得注意的是，所有这些额外功能都 end-of-life保证会获得任何进一步的安全更新。

PHP AL2023 年的模块

AL2023 包括PHP核心中包含的许多PHP模块。 AL2023 并不打算将所有软件包都包含在PHP扩展社区库 (PECL) 中。

AL2023 中的 Python

AL2023 移除了 Python 2.7，任何需要 Python 的组件现在均已改写为与 Python 3 配合工作。

AL2023 将 Python 3 作为 `/usr/bin/python3` 提供，以保持与客户代码以及 AL2023 随附的 Python 代码的兼容性，这在 AL2023 的整个生命周期内将保持为 Python 3.9。

`/usr/bin/python3` 所指向的 Python 版本被视为系统 Python，对于 AL2023，此为 Python 3.9。

更新版本的 Python，例如 Python 3.11，在 AL2023 中作为程序包提供，并在上游版本的生命周期内受支持。有关 Python 3.11 支持时长的信息，请参阅 Python 3.11。

可以在 AL2023 上同时安装 Python 的多个版本。尽管 `/usr/bin/python3` 将始终是 Python 3.9，但每个 Python 版本都是命名空间的，可以通过其版本号找到。例如，如果安装了 `python3.11`，则 `/usr/bin/python3.11` 将与 `/usr/bin/python3.9` 并存，而 `/usr/bin/python3` 符号链接指向 `/usr/bin/python3.9`。

> **Note**
>
> 请勿更改 `/usr/bin/python3` 符号链接所指向的目标，因为这可能会破坏 AL2023 的核心功能。

AL2023 中的 Python 模块

各种 Python 模块在 AL2023 中被打包为 RPM。通常，仅针对 Python 的系统版本构建 Python 模块的 RPM。

Ruby在 AL2023

Rubyin AL2023 由 3.2 和 3.4 版本表示。Amazon Linux 遵循上游支持计划，任何Ruby版本的支持状态都可以随时在 Ruby 网站上查看。所有受支持的 Ruby 版本都是命名空间化的，并且可以同时安装在同一系统上。命名空间化确保每个 Ruby 安装在文件系统内是唯一的。这是通过根据运行时版本重命名关键目录和文件来实现的。实际的可执行文件名称将看起来像 r uby {MAJOR.MINOR}（例如，`ruby3.2`或`ruby3.4`）。 Ruby3.4 还提供了 MRI（Matz's Inter Ruby preter）命名空间二进制文件`ruby3.4-mri`，它指的是基于 C 的标准参考实现。Ruby但是，一次只能有一个 Ruby 版本处于活动状态。此活动版本提供了默认目录和文件名，例如 ruby、gem 或 bundle，将它们指向当前活动的运行时。

这是利用 alternatives 工具的功能实现的。重要的是要记住，默认的可执行文件名是虚拟的，并且在指向不同安装的 Ruby 版本时可以随时更改。这种灵活性使在 shebang 中使用 ruby 的软件能够在调用时选择所需的版本。但是，当需要特定版本的Ruby时，可以通过调用命名空间可执行文件（例如`ruby3.2`或`ruby3.4`）来实现版本的持久性，该可执行文件将始终使用指定的运行时版本。此外，无论当前处于活动状态的运行时如何，gem 和捆绑器工具的命名空间可执行文件（例如`ruby3.4-bundler`、、或）始终与相应的Ruby版本相关联。`ruby3.2-gem` `ruby3.4-gem` `ruby3.2-bundler`

Ruby以几个以 “” `ruby{MAJOR.MINOR}` 开头的命名空间包的形式分发。这些软件包提供了 ruby、兼容版本的 gem 和捆绑器工具、文档、库等。例如，core Ruby 3.2 运行时由`ruby3.2`软件包提供，它作为依赖项引入`ruby3.2-rubygems`（提供 gem）和`ruby3.2-rubygem-bundler`（提供捆绑包和捆绑器）。

安装Ruby版本后，配套工具的条目可能会在备选配置中显示为空。这可以通过运行来验证`alternatives --display ruby`。如果条目显示为空，则必须使用手动注册`alternatives --install`。例如，要注册 Ruby 3.4 版的所有配套工具，请执行以下操作：

```
sudo alternatives --install /usr/bin/gem gem /usr/bin/ruby3.4-gem 34
sudo alternatives --install /usr/bin/bundle bundle /usr/bin/ruby3.4-bundle 34
sudo alternatives --install /usr/bin/bundler bundler /usr/bin/ruby3.4-bundler 34
sudo alternatives --install /usr/bin/erb erb /usr/bin/ruby3.4-erb 34
sudo alternatives --install /usr/bin/racc racc /usr/bin/ruby3.4-racc 34
sudo alternatives --install /usr/bin/rdoc rdoc /usr/bin/ruby3.4-rdoc 34
```

```
sudo alternatives --install /usr/bin/ri ri /usr/bin/ruby3.4-ri 34
```

优先级值（例如，3.4 为 34，Ruby3. Ruby 2 为 32）应与主红宝石替代条目中使用的优先级相匹配。注册后，配套工具将与 ruby 替代工具一起自动管理。

alternatives 工具提供了一个用于在 Ruby 版本之间切换的单一命令。默认情况下，alternatives 被配置为处于自动模式，该模式使用优先级来确定当前活动的 Ruby 版本。然而，您可以随时激活任何已安装的版本。目前，所有受支持的 Ruby 版本具有相等的优先级，这意味着第一个安装的版本将自动激活。

使用 alternatives 的一些有用示例

1. 检查 alternatives 的配置内容

   ```
   alternatives --list
   ```

2. 检查 ruby 的当前配置

   ```
   alternatives --display ruby
   ```

3. 交互式更改 Ruby 版本

   ```
   alternatives --config ruby
   ```

4. 切换到手动模式并选择特定版本

   ```
   alternatives --set ruby /usr/bin/ruby{MAJOR.MINOR}
   ```

5. 切换回自动版本选择模式

   ```
   alternatives --auto ruby
   ```

Rust在 AL2023

您可能想构建在 Amazon Linux Rust上编写的代码，也可能需要使用随附 AL2023的工具链。

类似于 AL2， AL2023 将在操作系统的整个生命周期中更新Rust工具链。这或许是为了回应我们发布的工具链中的任何 CVE，或许是季度发布的一部分。

Rust 是一种发展速度相对较快的语言，大概每六周发布一个新版本。这些版本中可能添加了新的语言或标准库功能。尽管 AL2023 将在Rust工具链的生命周期中加入新版本，但这不会与上游Rust版本保

持一致。因此，如果您想使用Rust语言的尖端功能构建Rust代码，则 AL2023 可能不适合使用中提供的Rust工具链。

在的生命周期内 AL2023，不会从存储库中删除旧的软件包版本。如果需要旧的 Rust 工具链，您可以选择放弃新 Rust 工具链的错误修复和安全补丁，并使用适用于任何 RPM 的相同机制从存储库安装旧版本。

如果你想在上面构建自己的Rust代码 AL2023，你可以使用中包含的Rust工具链，因为你知道这个工具链可能会在 AL2023 整个生命周期中向前发展。 AL2023

AL2023 写入的 Lambda 函数 Rust

由于 Rust 编译为原生代码，Lambda 将 Rust 视为自定义运行时。您可以使用`provided.al2023`运行时将Rust函数部署 AL2023 到 Lambda 上。

有关更多信息，请参阅《AWS Lambda 开发人员指南》中的使用 Rust 构建 Lambda 函数。

AL2023 中的 TypeScript

> **Note**
>
> 本文档提供了关于 TypeScript 及其基于 Node.js 的执行环境的基本信息。它还涵盖了典型的开发工作流程，并解释了 TypeScript 如何打包在 AL2023 中以提供一致且可复现的开发环境。

TypeScript（TS）是一种基于 JavaScript（JS）的编程语言，它提供 JS 的所有功能，并且还通过类型系统对其进行了扩展。在典型场景中，用 TS 编写的程序首先被翻译成 JS 代码，然后像任何其他常规 JS 程序一样由 Node.js 执行。在 TS 的特定术语中，此翻译过程被称为编译，并由称为 tsc 的编译器执行。tsc 编译器本身是用 JS 编写的，因此要运行它，也需要一个 JS 运行时环境，例如 Node.js。与其他一些 JS 运行时环境不同，Node.js 目前仅提供实验性和轻量级的 TS 支持。完整的 TS 支持，包括类型检查，仍然需要使用第三方程序包，例如 typescript。获取 Node.js 运行时环境的 tsc（TS 编译器）的预期方式是安装 typescript node 模块。这可以使用程序包管理器之一来完成，通常是 npm。使用 npm 安装 TS 编译器有两种方式：全局安装和在项目中安装。官方推荐的方法是基于每个项目安装 TS 编译器，这确保了项目的长期一致性和可复现性。然而，全局安装 TS 编译器可能仍然有用，因为它为整个主机及其 JS 运行时提供相同的版本，从而适用于本地未安装 TS 编译器的项目。这正是 Amazon Linux 上可用的 RPM 程序包（例如 `nodejs20-typescript` 或 `nodejs22-typescript`）安装 TS 编译器的方式：在系统级别全局安装，并为每个受支持的 Node.js 版本单独安装。

tsc 不直接依赖于任何 Node.js 版本。编译器期望一定级别的运行时功能，这些功能通过选项（如 target 和 lib）在特殊文件（tsconfig.json）中定义。这些选项的值代表 ECMAScript（ES）标准的一个版本，该版本可能（或可能不）被 JS 运行时环境支持。不同版本的 Node.js 支持不同版本的 ES 标准。Node.js 的版本越新，支持的 ES 标准版本越高且越完整。如果项目的根目录中不存在 tsconfig.json，则将使用默认的配置选项集。不同版本 Node.js 的兼容性表以及各种 ES 标准版本的支持特性可在 node.green 找到。tsc 有超过 100 个不同的选项，这些选项可以在 tsconfig.json 中定义。还支持配置链，即某些配置选项在另一个文件中定义，然后包含在主文件中。这种方法允许安装与特定版本 Node.js 兼容的基础 TS 配置，然后使用项目特定选项进行扩展。幸运的是，适用于 Node.js 的基础 TS 配置可作为 node 模块使用，可以使用 npm 将其安装在项目文件夹中。这是适用于 Node.js 版本 18、20 和 22 的配置源代码。

基于 Node.js 的运行时设计存在一个弱点：它在一台主机上仅支持一个运行时版本，并且要求在项目级别所有依赖项具有可复现性和一致性。这导致了以下使用 TypeScript 的常见方法：TS 编译器、当前 Node.js 版本的 TS 基础配置以及所有软件依赖项都在项目内部本地安装。虽然全局安装的 node 模块预期仅为 CLI 工具，例如 npm，但同样作为 CLI 工具的 tsc 却很少被全局安装。值得庆幸的是，tsc 的全局（系统范围内）和本地（项目内）安装可以毫无问题地共存，并且可以是独立使用的不同版本。请注意，本地安装的 tsc 应使用与 npm 一起安装的 npx 工具来执行。因此，即使有系统 TS 编译器，用户也有机会选择运行时组件的版本，例如 Node.js（通过 alternatives 切换活动版本）、TS 编译器（通过本地安装，或者全局安装并通过 alternatives 切换活动版本），并根据特定需求对其进行配置。

Amazon Linux 打包 TS 编译器的方式与其他全局安装的 node 模块（例如 npm）相同，基于每个 Node.js 版本进行。程序包和二进制文件均采用命名空间组织方式，并且在其名称中包含 Node.js 的主版本号。编译器的默认可执行文件名称 tsc 在运行时由 alternatives 工具管理，并指向当前活动的、为其安装并将由其执行的 Node.js 版本。此选择不依赖于当前的 Node.js 运行时版本。可能会出现 node 可执行文件指向 Node.js 20，而 tsc 被配置为由 Node.js 22 解释的情况。也可以使用 TS 编译器的命名空间名称，例如 tsc-{MAJOR_VERSION}，这与默认的 tsc 名称配置为何无关。

用于管理 TS 编译器活动版本的一些有用命令

1. 检查 alternatives 的配置内容

```
alternatives --list
```

2. 检查 tsc 的当前配置

```
alternatives --display tsc
```

3. 交互式更改 tsc 版本

```
alternatives --config tsc
```

4. 切换到手动模式并选择特定版本

```
alternatives --set tsc /usr/bin/tsc-{MAJOR_VERSION}
```

5. 切换回自动版本选择模式

```
alternatives --auto tsc
```

在同一系统上安装和使用多个 Node 版本及 TS 编译器的示例：

```
# 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 保留用户和组

AL2023 在映像配置期间以及安装某些程序包期间会预分配特定的用户和组。此处列出了用户、组及其关联的 UID 和 GID，以防止冲突。

主题

AL2023 保留用户列表

用户名称	UID
根	0
bin	1
daemon	2
adm	3
lp	4
同步	5
shutdown	6
停止	7
邮件	8
operator	11
游戏	12
ftp	14
squid	23

用户名称	UID
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
postfix	89

用户名称	UID
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
memcached	976

用户名称	UID
sphinx	977
haproxy	978
flatpak	979
debuginfod	980
dovenull	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
systemd-coredump	996

用户名称	UID
libstoragemgmt	997
systemd-oom	999
ec2-user	1000
nobody	65534

按名称列出

用户名称	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
daemon	2

用户名称	UID
dbus	81
debuginfod	980
dnsmasq	982
dovecot	97
dovenull	981
ec2-instance-connect	993
ec2-user	1000
flatpak	979
ftp	14
游戏	12
停止	7
haproxy	978
ldap	55
libstoragemgmt	997
lp	4
邮件	8
mailnull	47
memcached	976
munge	975
mysql	27

用户名称	UID
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
根	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179

用户名称	UID
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
同步	5
systemd-coredump	996
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
tcpdump	72

用户名称	UID
Tomcat	53
tss	59
unbound	983
uuidd	961

AL2023 保留组列表

Group name	GID
根	0
bin	1
daemon	2
sys	3
adm	4
tty	5
磁盘	6
磁盘	6
lp	7
mem	8
kmem	9
wheel	10
cdrom	11

Group name	GID
邮件	12
邮件	12
man	15
dialout	18
floppy	19
游戏	20
slocate	21
utmp	22
squid	23
named	25
postgres	26
mysql	27
nscd	28
nscd	28
rpcuser	29
rpc	32
磁带	33
utempter	35
kvm	36
视频	39

Group name	GID
mailnull	47
Apache	48
ftp	50
smmsp	51
Tomcat	53
锁定	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
dovecot	97

Group name	GID
用户	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
wireshark	960

Group name	GID
uuidd	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
跟踪	973
haproxy	974
flatpak	975
debuginfod	976
dovenull	977
dnsmasq	978
unbound	979
clamscan	980

Group name	GID
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
libstoragemgmt	997
ssh_keys	998
systemd-oom	999

Group name	GID
ec2-user	1000
ne	1001
polkitd	9920
nobody	65534

按名称列出

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
colord	985

Group name	GID
cwagent	989
daemon	2
dbus	81
debuginfod	976
dialout	18
磁盘	6
磁盘	6
dnsmasq	978
Docker	987
dovecot	97
dovenull	977
ec2-instance-connect	993
ec2-user	1000
flatpak	975
floppy	19
ftp	50
游戏	20
haproxy	974
input	104
kmem	9

Group name	GID
kvm	36
ldap	55
libstoragemgmt	997
锁定	54
lp	7
邮件	12
邮件	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
nscd	28

Group name	GID
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
根	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179
saslauth	76
screen	84

Group name	GID
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
systemd-journal-remote	963

Group name	GID
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
磁带	33
tcpdump	72
Tomcat	53
跟踪	973
tss	59
tty	5
unbound	979
usbmon	959
用户	100
usershares	964
utempter	35
utmp	22
uuidd	961
视频	39
virusgroup	982
virusgroup	982

Group name	GID
virusgroup	982
wbpriv	88
wheel	10
wireshark	960

2023 年可用的编解码器列表 AL2

AL2023 通过其标准存储库提供了一系列多媒体编解码器。本页概述了编解码器及其典型使用案例。

> **Important**
>
> 使用和分发 Amazon Linux 中包含的编解码器可能要求您从第三方获得许可证权利，包括某些第三方音频和视频格式的所有者或许可方。您全权负责获取这些许可证并支付所有必要的版税或费用。

编解码器	说明
flac	一款免费开源的无损音频编解码器，可在压缩音频时不损失任何数据或质量，常用于高质量音频存储
fdk-aac-free	AAC（高级音频编解码器）标准的开源实现，为流媒体或文件存储等 MP3 替代方案提供高质量的音频压缩
webrtc-audio-processing	用于 WebRTC（网页实时通信）的音频处理库，提供噪声抑制、回声消除和增益控制等功能
opus	一款专为实时流媒体设计的高度通用且高效的音频编解码器，具有低延迟特性，支持包括 VoIP 和音乐流媒体在内的多种音频应用
libsndfile	用于读写多种格式（如 WAV、AIFF 和 FLAC）音频文件的库，常用于音频处理和编辑工具
openh264	H.264 视频编解码器的开源实现。它为流媒体、会议和多媒体应用程序中使用的 H.264 视频提供编码和解码支持。
svt-av1	AV1 视频编解码器的开源、高性能实现。它为 AV1 视频提供可扩展的编码和解码，并针对现代 CPUs 和并行处理进行了优化。

编解码器	说明
dav1d	AV1 视频解码器的开源实现侧重于速度、效率和便携性。它为各种平台和应用程序提供高性能 AV1 解码。
libde265	H.265/HEVC 视频解码器的开源实现。它为 HEVC 视频流提供高效、便携的解码，用于多媒体应用程序和框架。
libheif	一个用于读取和写入 HEIF 和 AVIF 图像文件的开源库。它使用现代压缩格式（如 HEVC 和）为图像和图像序列提供高效的编码、解码和转换。 AV1
mpg123	适用于包括 MP3 文件在内的 MPEG 音频流的高性能解码器。它为播放、流媒体和音频处理应用程序提供快速、准确的音频解码。
x265	x265 的主要目标是成为任何地方可用的最佳 H.265/HEVC 编码器，在各种硬件平台上提供最高的压缩效率和最高的性能。该软件包包含命令行编码器。

Amazon Linux 2023 中的安全性与合规性

> **Important**
>
> 如果您想报告漏洞或对 AWS 云服务或开源项目有安全方面的问题，请使用漏洞报告页面联系 AWS 安全部门。

AWS 十分重视云安全性。作为 AWS 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是 AWS 和您的共同责任。责任共担模型在此描述为云的安全性和云中的安全性：

- 云安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础结构。AWS 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 AWS 合规性计划的一部分。要了解适用于 AL2023 的合规性计划，请参阅 AWS 按合规性计划提供的范围内服务。
- 云中的安全性：您的责任由您使用的 AWS 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

主题

亚马逊 Linux 安全公告 AL2023

尽管我们努力确保 Amazon Linux 的安全，但有时会有一些安全问题需要修复。当有可用的修复程序时，我们就会发布通告。我们发布通告的主要位置是 Amazon Linux 安全中心（ALAS）。有关更多信息，请参阅 Amazon Linux 安全中心。

> **Important**
>
> 如果您想报告漏洞或对 AWS 云服务或开源项目有安全疑虑，请使用 “漏洞报告” 页面与 AWS 安全部门联系

Amazon Linux 团队在多个地点发布了有关问题和相关更新的信息。 AL2023 安全工具通常会从这些主要源代码中获取信息并将结果呈现给您。因此，您可能不会直接与 Amazon Linux 发布的主要源代码进行交互，而是与您的首选工具（例如 Amazon Inspector）提供的界面进行交互。

Amazon Linux 安全中心公告

Amazon Linux 公告用于不适合放入通告的项目。这部分包含关于 ALAS 本身的公告，以及不适合放入通告的信息。有关更多信息，请参阅 Amazon Linux Security Center（ALAS）公告。

例如，2021-001 号关于 Apache Log4j 的 Amazon Linux 热补丁公告应归类为公告而非通告。在此公告中，Amazon Linux 添加了一个程序包以帮助客户缓解不属于 Amazon Linux 的软件中的安全问题。

Amazon Linux Security Center CVE Explorer 也通过 ALAS 公告宣布推出。有关更多信息，请参阅新网站 CVEs。

Amazon Linux 安全中心常见问题解答

有关 ALAS 和 Amazon Linux 如何评估的一些常见问题的答案 CVEs，请参阅亚马逊 Linux 安全中心 (ALAS) 常见问题解答 ()。FAQs

ALAS 通告

Amazon Linux 通告包含与 Amazon Linux 用户相关的重要信息，通常是关于安全更新的信息。可以通过 Amazon Linux 安全中心在网络上查看通告。通告信息也是 RPM 程序包存储库元数据的一部分。

通告和 RPM 存储库

Amazon Linux 2023 程序包存储库可能包含描述零个或多个更新的元数据。`dnf updateinfo` 命令以包含此信息的存储库元数据文件名 `updateinfo.xml` 命名。虽然命令名为 `updateinfo`，而元数据文件引用 `update`，但这些都指代属于通告一部分的程序包更新。

Amazon Linux 通告发布在 Amazon Linux 安全中心网站，同时信息也存在于 dnf 程序包管理器引用的 RPM 存储库元数据中。网站和存储库元数据最终会保持一致，但网站和存储库元数据中的信息可能存在暂时性不一致的情况。这通常发生在发布新版本时，在最新 AL2023 版本之后公告有更新时。AL2023

虽然新通告通常与解决问题的程序包更新一起发布，但并非总是如此，也可以为已在已发布程序包中解决的新问题创建通告。现有公告也可以使用新公告进行更新 CVEs ，现有更新将解决这些问题。

Amazon Linux 2023 的通过版本控制的存储库进行确定性升级 AL2023功能意味着特定 AL2023 版本的 RPM 存储库包含截至该版本的 RPM 存储库元数据的快照。这包括描述安全更新的元数据。特定 AL2023 版本的 RPM 存储库在发布后不会更新。查看旧版本的 AL2023 RPM 存储库时，新的或更新的安全公告将不可见。有关如何使用`dnf`软件包管理器查看`latest`存储库版本或特定 AL2023 版本的信息，请参阅列出适用的通告部分。

咨询 IDs

每个通告由一个 `id` 引用。目前，这是亚马逊 Linux 的一个怪癖，亚马逊 Linux 安全中心网站将公ALAS2023告列为 ALAS-2024-581，而`dnf`软件包管理器会将该公告列为 -2024-581。当 原地应用安全更新 时，程序包管理器 ID 需要用于引用特定通告。

对于 Amazon Linux，每个主要版本的操作系统都有自己的咨询命名空间 IDs。不应对 Amazon Linux 公告的格式做出任何假设 IDs。从历史上看，Amazon Linux Advisory IDs 一直遵循以下模式`NAMESPACE-YEAR-NUMBER`。`NAMESPACE` 的可能值范围未定义，但包括 `ALAS`、`ALASCORRETTO8`、`ALAS2023`、`ALAS2`、`ALASPYTHON3.8` 和 `ALASUNBOUND-1.17`。`YEAR` 是创建通告的年份，`NUMBER` 是命名空间内的唯一整数。

尽管 IDs Ad visory 通常是按更新发布顺序排列的，但有许多原因不可能出现这种情况，因此不应假设这种情况。

将通告 ID 视为对每个 Amazon Linux 主要版本唯一的不透明字符串。

在 Amazon Linux 2 中，每个 Extra 在单独的 RPM 存储库中，通告元数据仅包含在相关的存储库中。一个存储库的通告不适用于另一个存储库。在 Amazon Linux 安全中心网站上，目前每个 Amazon Linux 主要版本都有一个通告列表，并未按存储库列表分开。

由于 AL2023 不使用 Extras 机制来打包其他版本的软件包，因此目前只有两个 RPM 存储库，每个存储库都有咨询、core存储库和存储livepatch库。livepatch 存储库用于 AL2023 上的内核实时修补。

通告发布日期和通告更新日期

Amazon Linux 通告的通告发布日期表示安全更新首次在 RPM 存储库中公开发布的时间。修复方案发布至 RPM 存储库可供安装后，相关通告会立即在 Amazon Linux 安全中心网站发布。

通告更新日期表示在先前发布后向通告添加新信息的时间。

不应在 AL2023 版本号（例如 2023.6.20241031）和与该版本一起发布的公告公告发布日期之间做出任何假设。

通告类型

RPM 存储库元数据支持不同类型的通告。虽然 Amazon Linux 几乎只发布安全更新的通告，但不应假设情况始终如此。可能发布用于错误修复、增强功能和新程序包等事件的通告，且通告标记为包含此类更新。

通告严重性

每份通告有自己的严重性，因为每个问题单独评估。在单个通告中 CVEs 可以解决多个问题，每个 CVE 可能有不同的评估，但该通告本身具有一个严重性。可能有多份通告引用单个程序包更新，因此特定程序包更新可能有多个严重性（每份通告一个）。

按严重性递减顺序，Amazon Linux 使用严重、重要、中等和低来表示通告的严重性。Amazon Linux 通告也可能没有严重性，尽管这极其罕见。

Amazon Linux 是使用术语“中等”的基于 RPM 的 Linux 发行版之一，而其他一些基于 RPM 的 Linux 发行版使用等效术语“中”。Amazon Linux 程序包管理器将两个术语视为等效，且第三方程序包存储库可能使用术语“中”。

随着对通告中解决的相关问题了解更多，Amazon Linux 通告可以随时间改变严重性。

通告的严重性通常会跟踪该公告所 CVEs 引用的 Amazon Linux 评估的最高 CVSS 分数。在某些情况下可能不是这样。例如，某些已修复但未分配 CVE 编号的问题便会采用此种处理方式。

更多关于 Amazon Linux 如何使用通告严重性评级的信息，请参阅 ALAS 常见问题解答。

通告和程序包

单个程序包可能有多份通告，且并非所有程序包都会有通告发布。可以在多个公告中引用特定的软件包版本，每个公告都有自己的严重性和。 CVEs

在一个新版本中可以同时发布多个针对同一个软件包更新的公告，也可以快速连续 AL2023 发布。

像其他 Linux 发行版一样，从一个源程序包可以构建一个到多个不同的二进制程序包。例如，ALAS-2024-698 是亚马逊 Linux 安全中心网站AL2023 部分列出的适用于该`mariadb105`软件包的公告。这是源程序包名称，通告本身引用二进制程序包以及源程序包。在这种情况下，从一个 `mariadb105` 源程序包构建了十几个二进制程序包。虽然通常有一个与源程序包同名的二进制程序包，但这并不普遍。

虽然 Amazon Linux 通告通常列出从更新的源程序包构建的所有二进制程序包，但不应假设情况始终如此。程序包管理器和 RPM 存储库元数据格式允许通告列出更新的二进制程序包子集。

特定通告也可能仅适用于特定 CPU 架构。有些程序包可能不是为所有架构构建的，也可能存在不影响所有架构的问题。在程序包在所有架构上可用但问题仅适用于一个架构的情况下，Amazon Linux 通常不会发布仅引用受影响架构的通告，尽管不应假设情况始终如此。

由于程序包依赖关系的性质，通常通告引用一个程序包，但安装该更新需要其他程序包更新，包括未在通告中列出的程序包。`dnf` 程序包管理器将处理安装所需的依赖项。

公告和 CVEs

一份公告可能涉及零个或多个 CVEs，并且可能有多个咨询引用同一 CVE。

建议何时可能引用零的一个例子 CVEs 是尚未为问题分配 CVE（或从来没有）。

多份通告可能引用同一 CVE 的例子是当 CVE 适用于多个程序包时。例如，CVE-2024-21208 适用于 Corretto 8、11、17 和 21。这些 Corretto 版本 AL2023中的每一个都是一个单独的软件包，每个软件包都有公告：ALAS-2024-754 适用于 Corretto 8，ALAS-2024-753 适用于 Corretto 11，ALAS-2024-752 适用于 Corretto 17，ALAS-2024-752 适用于 Corretto 21。虽然这些 Corretto 版本都有相同的列表，但不应 CVEs假设这一点。

对于不同的程序包，可以对特定 CVE 进行不同的评估。例如，如果特定 CVE 在通告中以“重要”严重性引用，可能发布另一份通告引用同一 CVE 但严重性不同。

RPM 存储库元数据允许列出每份通告的参考文献。虽然 Amazon Linux 通常只引用 CVEs，但元数据格式确实允许其他参考类型。

RPM 软件包存储库元数据将仅在 CVEs 有可用的修复程序时引用。亚马逊 Linux 安全中心网站的 “浏览” 部分 CVEs 包含有关亚马逊 Linux 评估过的信息。此评估可能导致针对各种 Amazon Linux 版本和程序包的 CVSS 基本分数、严重性和状态。特定 Amazon Linux 版本或程序包的 CVE 状态可能为“不受影响”、“待修复”或“无修复计划”。在公告发布之前，的状态和评估 CVEs 可能会多次以任何方式发生变化。这包括重新评估 CVE 对 Amazon Linux 的适用性。

该公告首次发布后，该公 CVEs 告所引用的列表可能会发生变化。

通告文本

通告还将包含描述导致创建通告的问题的文本。通常此文本是未修改的 CVE 文本。此文本可能引用上游版本号，其中修复程序可用，与 Amazon Linux 已应用修复程序的程序包版本不同。通常 Amazon Linux 会从较新的上游版本回溯修复程序。在通告文本提及的上游版本与 Amazon Linux 版本中提供的版本不同的情况下，通告中的 Amazon Linux 程序包版本对于 Amazon Linux 来说将是准确的。

RPM 存储库元数据中的通告文本可能是占位符文本，详情请访问 Amazon Linux 安全中心网站。

内核实时补丁通告

实时补丁通告的独特之处在于它们引用不同的程序包（Linux 内核）而不是通告针对的程序包（例如 `kernel-livepatch-6.1.15-28.43`）。

内核动态补丁公告将引用特定的 Live Patch 包可以针对该实时补丁包所适用的特定内核版本解决的问题（例如 CVEs）。

每个实时补丁都适用于特定内核版本。为了应用 CVE 的实时补丁，需要安装适用于您内核版本的正确实时补丁程序包，并应用实时补丁。

例如，可以针对 AL2023 内核版本、和 `6.1.56-82.125` `6.1.59-84.139`6.1.61-85.141，对 CVE-2023-6111 进行实时修补。还发布了包含此 CVE 修复程序的新内核版本，并有单独的通告。要在等于或高于 ALAS2023-2023-461 所指定版本的内核版本上 AL2023 解决 CVE-2023-6111 问题，或者需要在应用适用的 livepatch 的情况下运行带有此 CVE 实时补丁的内核版本之一。

当有新的实时补丁可用于特定内核版本时（该版本已有实时补丁可用），会发布新版本的 `kernel-livepatch-KERNEL_VERSION` 程序包。例如，发布公ALASLIVEPATCH-2023-003告的`kernel-livepatch-6.1.15-28.43-1.0-1.amzn2023`软件包中包含了涵盖三个`6.1.15-28.43`内核的实时补丁 CVEs。后来，该`kernel-livepatch-6.1.15-28.43-1.0-2.amzn2023`软件包发布了公ALASLIVEPATCH-2023-009告；这是对先前`6.1.15-28.43`内核的实时补丁包的更新，其中包含另外三个的实时补丁 CVEs。还有其他针对其他内核版本的实时补丁通告问题，程序包中包含针对那些特定内核版本的实时补丁。

更多关于内核实时补丁的信息，请参阅 AL2023 上的内核实时修补。

对于任何围绕安全通告开发工具的人员，还建议查看 通告和 updateinfo.xml 的 XML 模式 部分以获取更多信息。

通告和 updateinfo.xml 的 XML 模式

updateinfo.xml 文件是程序包存储库格式的一部分。它是 dnf 程序包管理器解析以实现功能（如列出适用的通告 和 原地应用安全更新）的元数据。

我们建议使用 dnf 程序包管理器的 API，而不是编写自定义代码来解析存储库元数据格式。dnfin 的版本 AL2023可以解析 AL2023 和 AL2 存储库格式，因此 API 可用于检查任一操作系统版本的咨询信息。

RPM 软件管理项目在 rpm 元数据存储库中记录了 RPM 元数据格式。 GitHub

对于围绕直接解析 updateinfo.xml 元数据开发工具的人员，强烈建议仔细关注 rpm-metadata 文档。该文档涵盖了实际应用中遇到的情况，其中包含许多可能被您合理理解为元数据格式规则的例外情况。

上的 raw-historical-rpm-repository-examples 存储库中还有越来越多的真实updateinfo.xml文件示例。 GitHub

如果文档中有任何不清楚的地方，你可以在 GitHub 项目上打开一个问题，这样我们就可以回答问题并适当地更新文档。作为开源项目，我们也欢迎通过拉取请求来贡献文档更新。

列出适用的通告

dnf 程序包管理器可以访问描述哪些程序包版本修复了哪些通告的元数据。因此，它可以列出哪些通告适用于某个实例或容器映像。

> **Note**
>
> 诸如 AWS Systems Manager 之类的工具可以利用此功能来显示整个实例集而不仅仅是单个实例相关的更新。

在列出更新时，您可以指示 dnf 查看特定 AL2023 发布的元数据，或来自最新发布的元数据。

> **Note**
>
> AL2023 发布一旦完成，即不可变。因此，Amazon Linux 安全中心上的新的或更新的通告只会添加到 AL2023 新发布的元数据中。

我们现在将通过示例来了解哪些通告适用于某些 AL2023 容器映像。这些命令在非容器化环境（例如 EC2 实例）中均可使用。

Listing advisories in a specific version

在此示例中，我们将查看 2023.1.20230628 发布中有哪些通告与 2023.0.20230315 发布的容器映像相关。

> **Note**
>
> 此示例使用 2023.0.20230315 和 2023.1.20230628 发布，这些并非 AL2023 的最新发布。有关包含最新安全更新的最新发布，请参阅 AL2023 发布说明。

在此示例中，我们将从 2023.0.20230315 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 `.0` 表示特定发布的映像版本；此映像版本通常为零。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中要求 `dnf` 列出与容器中安装的程序包相关的通告。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
   bash-5.2#
```

现在使用 `dnf updateinfo` 命令来显示 2023.1.20230628 发布中哪些通告与我们安装的程序包相关的摘要。

```
$ 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)
```

要获取通告列表，可以向 `dnf updateinfo` 提供 `--list` 选项。

```
$ 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

在此示例中，我们将查看如果启动了一个 2023.4.20240319 发布的容器，在 AL2023 的 `latest` 版本中有哪些更新可用。在撰写本文时，`latest` 发布为 2023.5.20240708，因此此示例中列出的更新将截至该发布。

> **Note**
>
> 此示例使用 2023.4.20240319 和 2023.5.20240708 发布，后者是在撰写本文时的最新发布。有关最新发布的更多信息，请参阅 AL2023 发布说明。

在此示例中，我们将从 2023.4.20240319 发布的容器映像开始。

首先，我们从容器注册表获取此容器映像。末尾的 .1 表示特定发布的映像版本。虽然映像版本通常为零，但此示例使用的发布的映像版本为一。

```
$ 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
```

我们现在可以在容器内生成一个 shell，并从中检查更新。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
   bash-5.2#
```

现在使用 dnf updateinfo 命令来显示最新发布中哪些通告与我们安装的程序包相关的摘要。在撰写本文时，2023.1.20230628 是最新发布。

```
$ 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)
```

要获取通告列表，可以向 dnf updateinfo 提供 --list 选项。

```
$ 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
```

原地应用安全更新

有关应用更新的概述，请参阅 使用 DNF 和存储库版本应用安全更新。dnf upgrade 的 --security 选项将把程序包更新限制为仅那些有通告的程序包。这部分的剩余部分将介绍如何仅安装特定的安全更新。

> **Note**
>
> 建议应用新 AL2023 发布中可用的所有更新。仅选择安全更新或仅特定更新应是例外而非规则。

应用通告中提到的更新

dnf upgradeinfo 输出第一列中的通告标识符可用于应用通告中提到的程序包的更新。可以指示 dnf 程序包管理器将通告中的程序包更新至最新可用版本，或仅更新至通告中提到的版本。如果更新已安装，则更新命令不执行任何操作。

要仅将受影响程序包更新至通告中提到的版本，请在使用 dnf upgrade-minimal 命令时使用 --advisory 选项指定通告。以下示例在 AL2023 版本 2023.0.20230315 容器中运行 dnf upgrade-minimal。

```
$ 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!
```

即使使用 `--releasever=latest`，更新的程序包版本也是相同的，因为请求是让 dnf 执行解决通告所需的最小更新。

使用带有 `--advisory` 选项的常规 `dnf upgrade` 命令会将通告中提到的相关程序包更新至可用的最新版本，该版本可能比通告中提到的版本更新。

> **Note**
>
> 除非更新 `system-release` 程序包，否则 dnf 锁定的 AL2023 存储库版本不会改变。

> **Warning**
>
> 在安装来自不同 AL2023 发布的更新而不更改 dnf 锁定的存储库版本时，必须在后续的任何变更性 dnf 操作中小心谨慎。例如，在安装或更新程序包时，由于新发布中的程序包依赖关系可能已更改，您所停留的旧发布可能无法满足这些新的依赖关系。

以下示例在 AL2023 版本 2023.0.20230315 容器中运行，该容器引用了撰写本文时最新的 AL2023 发布 2023.5.20240708。请注意，curl 更新到的版本比 update-minimal 更新到的版本新，但这个新版本引入了新的依赖项。

```
$ 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!
```

为 AL2023 设置 SELinux 模式

默认情况下，安全增强型 Linux（SELinux）在 AL2023 中为 `enabled` 并设置为 `permissive` 模式。在许可模式下，记录但不强制执行权限拒绝。SELinux 是内核功能和实用程序的集合，可为内核的主要子系统提供强大、灵活的强制访问控制 (MAC) 架构。

SELinux 提供了一种增强的机制，可根据机密性和完整性要求强制信息分离。这种信息分离减少了篡改和绕过应用程序安全机制的威胁，也限制了恶意或有缺陷的应用程序可能导致的损害。

SELinux 包含一组示例安全策略配置文件，旨在满足日常安全目标。

有关 SELinux 特性和功能的更多信息，请参阅 SELinux Notebook 和策略语言。

主题

AL2023 的默认 SELinux 状态和模式

对于 AL2023，SELinux 默认状态为 enabled 并设置为 permissive 模式。在 permissive 模式下，记录但不强制执行权限拒绝。

getenforce 或 **sestatus** 命令会告诉您当前的 SELinux 状态、策略和模式。

当默认状态设置为 enabled 和 permissive 时，**getenforce** 命令返回 permissive。

sestatus 命令返回 SELinux 状态和当前 SELinux 策略，如下例所示：

```
$ 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
```

当您在 permissive 模式下运行 SELinux 时，用户可能会错误地标记文件。当您在 disabled 状态下运行 SELinux 时，不会标记文件。当您改为 enforcing 模式时，不正确或未标记的文件都可能导致问题。

SELinux 会自动重新标记文件以避免此问题。当您将状态改为 enabled 时，SELinux 可通过自动重新标记来防止出现标记问题。

改为 enforcing 模式

当您在 enforcing 模式下运行 SELinux 时，SELinux 实用程序就是 enforcing 配置的策略。SELinux 基于策略规则来允许或拒绝访问，以此来控制选定应用程序的功能。

要查找当前 SELinux 模式，请运行 getenforce 命令。

```
getenforce
Permissive
```

编辑配置文件以启用 enforcing 模式

您可以使用以下步骤将模式改为 enforcing。

1. 编辑 /etc/selinux/config 文件以改为 enforcing 模式。SELINUX 设置应类似以下示例。

   ```
   SELINUX=enforcing
   ```

2. 重新启动系统以完成改为 enforcing 模式。

   ```
   $ sudo reboot
   ```

在下次启动时，SELinux 会重新标记系统中的所有文件和目录。SELinux 还会为在 SELinux 处于 disabled 模式时创建的文件和目录添加 SELinux 上下文。

切换到 enforcing 模式后，由于 SELinux 策略规则不正确或缺失，SELinux 可能会拒绝某些操作。您可以使用以下命令查看 SELinux 拒绝的操作。

```
$ sudo ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

使用 cloud-init 启用 enforcing 模式

或者，当您启动实例时，请将以下 cloud-config 作为用户数据传递以启用 enforcing 模式。

```
#cloud-config
selinux:
  mode: enforcing
```

默认情况下，此设置会导致实例重新启动。为了提高稳定性，建议您重启实例。但是，如果您愿意，可以通过提供以下 cloud-config 来跳过重新启动。

```
#cloud-config
selinux:
  mode: enforcing
  selinux_no_reboot: 1
```

为 AL2023 禁用 SELinux 的选项

当您禁用 SELinux 时，SELinux 策略不会被加载或强制执行，并且访问向量缓存（AVC）消息不会被记录。您将失去运行 SELinux 的所有好处。

我们建议使用 permissive 模式，而不是禁用 SELinux。在 permissive 模式下运行的成本仅比完全禁用 SELinux 略高一点。从 permissive 模式过渡到 enforcing 模式所需的配置调整，远比在禁用 SELinux 后重新过渡回 enforcing 模式要少得多。您可以标记文件，系统可以跟踪和记录活动策略可能拒绝的操作。

将 SELinux 更改为 **permissive** 模式

当您在 permissive 模式下运行 SELinux 时，SELinux 策略不会被强制执行。在 permissive 模式下，SELinux 记录 AVC 消息但不拒绝操作。您可以使用这些 AVC 消息进行故障排除、调试和改进 SELinux 策略。

要将 SELinux 更改为宽容模式，请使用以下步骤。

1. 编辑 /etc/selinux/config 文件以改为 permissive 模式。SELINUX 值应类似以下示例。

   ```
   SELINUX=permissive
   ```

2. 重新启动系统以完成改为 permissive 模式。

   ```
   sudo reboot
   ```

禁用了 SELinux

当您禁用 SELinux 时，SELinux 策略不会被加载或强制执行，并且 AVC 消息不会被记录。您将失去运行 SELinux 的所有好处。

要禁用 SELinux，请使用以下步骤。

1. 确保已安装 grubby 程序包。

```
rpm -q grubby
grubby-version
```

2. 配置您的引导加载程序以将 selinux=0 添加到内核命令行。

```
sudo grubby --update-kernel ALL --args selinux=0
```

3. 重新启动系统。

```
sudo reboot
```

4. 运行 getenforce 命令以确认 SELinux 为 Disabled。

```
$ getenforce
Disabled
```

有关 SELinux 的更多信息，请参阅 SELinux 笔记本和 SELinux 配置。

在 AL2023 上启用 FIPS 模式

这部分介绍如何在 AL2023 上启用“美国联邦信息处理标准 (FIPS)”。有关 FIPS 的更多信息，请参阅：

- 美国联邦信息处理标准 (FIPS)
- 合规性常见问题：美国联邦信息处理标准

> **Note**
>
> 这部分介绍如何在 AL2023 中启用 FIPS 模式，但不包括 AL2023 加密模块的认证状态。

先决条件

- 现有 AL2023（AL2023.2 或更高版本）Amazon EC2 实例，可以访问互联网并下载所需软件包。有关启动 AL2023 Amazon EC2 实例的更多信息，请参阅使用 Amazon EC2 控制台启动 AL2023。
- 必须使用 SSH 或 AWS Systems Manager 连接到您的 Amazon EC2 实例。有关更多信息，请参阅连接到 AL2023 实例。

> **Important**
>
> FIPS 模式不支持 ED25519 SSH 用户密钥。如果您使用 ED25519 SSH 密钥对启动了 Amazon EC2 实例，则必须使用其他算法（例如 RSA）生成新密钥，否则您在启用 FIPS 模式后可能失去对实例的访问权限。更多信息，请参阅《Amazon EC2 用户指南》中的创建密钥对。

启用 FIPS 模式

1. 使用 SSH 或 AWS Systems Manager 连接到 AL2023 实例。
2. 确保系统是最新版本。有关更多信息，请参阅 在中管理软件包和操作系统更新 AL2023。
3. 确保已安装 `crypto-policies` 实用程序且为最新版本。

   ```
   sudo dnf -y install crypto-policies crypto-policies-scripts
   ```

4. 通过运行以下命令来启用 FIPS 模式。这将为 AL2023 常见问题解答中列出的模块系统范围内启用 FIPS 模式

   ```
   sudo fips-mode-setup --enable
   ```

5. 使用以下命令重新引导实例。

   ```
   sudo reboot
   ```

6. 要验证是否已启用 FIPS 模式，请重新连接到实例并运行以下命令。

   ```
   sudo fips-mode-setup --check
   ```

   以下示例输出显示已启用 FIPS 模式：

   ```
   FIPS mode is enabled.
   ```

在 AL2023 容器中启用 FIPS 模式

这部分说明如何在 AL2023 容器中启用联邦信息处理标准（FIPS）。有关 FIPS 的更多信息，请参阅：

- 美国联邦信息处理标准 (FIPS)

- 合规性常见问题：美国联邦信息处理标准

> **Note**
>
> 这部分记录如何在 AL2023 容器中启用 FIPS 模式。它不涉及 AL2023 加密模块的认证状态。

先决条件

- 现有 AL2023（AL2023.2 或更高版本）Amazon EC2 实例，可以访问互联网并下载所需软件包。有关启动 AL2023 Amazon EC2 实例的更多信息，请参阅使用 Amazon EC2 控制台启动 AL2023。
- 必须使用 SSH 或 AWS Systems Manager 连接到您的 Amazon EC2 实例。有关更多信息，请参阅连接到 AL2023 实例。

> **Important**
>
> `fips-mode-setup` 命令在容器内部将无法正常工作。请阅读以下步骤以在 AL2023 容器中正确配置 FIPS 模式。

在 AL2023 容器中启用 FIPS 模式

1. 必须首先在 AL2023 容器主机上启用 FIPS 模式。按照 在 AL2023 上启用 FIPS 模式 处的说明在主机上启用 FIPS 模式。
2. 使用 SSH 或 AWS Systems Manager 连接到您的 AL2023 容器主机实例。
3. 如果 AL2023 主机处于 FIPS 模式并且 `/proc/sys/crypto/fips_enabled` 可以从容器内部访问，则 FIPS 模式将在 AL2023 容器中自动启用。如果 `/proc/sys/crypto/fips_enabled` 的内容是 `0`，则表示 FIPS 未启用；值为 `1` 表示 FIPS 模式已启用。

   您可以通过在 AL2023 主机和容器上运行以下命令来验证 FIPS 是否已启用：

   ```
   cat /proc/sys/crypto/fips_enabled
   ```

4. 接下来，在容器内启用 FIPS 加密策略。有几种方法可以实现这一点，将在下面的选项中描述。请使用最适合您环境的选项。

   a. 使用 `update-crypto-policies` 命令在容器内手动启用 FIPS 加密策略：

```
# Run these commands inside the container
dnf install -y crypto-policies-scripts
update-crypto-policies --set FIPS
```

b. 在 AL2023 容器内创建 bind 挂载（这类似于 podman 在其他发行版中的工作方式）：

```
# 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. 也可以创建一个绑定挂载，以使 AL2023 容器与 AL2023 主机的加密策略相匹配。以下仅作为示例提供。如果容器和主机之间的加密策略及程序包版本存在不兼容的差异，此配置可能会导致问题：

```
sudo docker pull amazonlinux:2023
sudo docker run --mount type=bind,readonly,src=/etc/crypto-policies,dst=/etc/
crypto-policies -it amazonlinux:2023
```

5. 执行上述步骤后，您可以使用以下命令再次验证 FIPS 是否已在容器中启用：

```
$ cat /etc/crypto-policies/config
FIPS

$ cat /proc/sys/crypto/fips_enabled
1
```

在 AL2023 上切换 OpenSSL FIPS 提供程序

这部分说明如何在 AL2023 上切换 `latest` 与 `certified` OpenSSL FIPS 提供程序。

有关 FIPS 的更多信息，请参阅：

- 美国联邦信息处理标准 (FIPS)
- 合规性常见问题：美国联邦信息处理标准
- FedRAMP 加密模块选择与使用策略

> **Important**
>
> 在 AL2023.7 及更高版本中，默认的 OpenSSL FIPS 提供程序是 `openssl-fips-provider-latest` 程序包，该程序包会定期接收错误修复和安全更新。
> 以下说明仅适用于希望锁定到 `openssl-fips-provider-certified` 程序包的客户。此版本 FIPS 提供程序的校验和将与 NIST 证书保持一致，但可能不包含最新更新。
> 有关 FIPS 认证模块和程序包版本的更多信息，请参阅 AL2023 常见问题解答。

先决条件

- 现有 AL2023（AL2023.7 或更高版本）Amazon EC2 实例，可以访问互联网并下载所需软件包。有关启动 AL2023 Amazon EC2 实例的更多信息，请参阅使用 Amazon EC2 控制台启动 AL2023。
- 必须使用 SSH 或 AWS Systems Manager 连接到您的 Amazon EC2 实例。有关更多信息，请参阅连接到 AL2023 实例。
- 要在 AL2023 上启用 FIPS 模式，请按照 在 AL2023 上启用 FIPS 模式 处的说明操作。

在 **openssl-fips-provider-latest** 和 **openssl-fips-provider-certified** 之间切换。

1. 使用 dnf 切换 OpenSSL FIPS 提供程序：

   ```
   sudo dnf -y swap openssl-fips-provider-latest openssl-fips-provider-certified
   ```

2. 验证当前正在使用经认证的 OpenSSL FIPS 提供程序。在启用 FIPS 模式的 AL2023 系统中运行以下命令：

   ```
   openssl list -providers
   ```

   您应看到以下输出：

   ```
   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 内核强化

中的 AL2023 6.1 Linux 内核配置和构建时有多个强化选项和功能。

内核强化选项（与架构无关）

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_ACPI_CUSTOM_METHOD	n	n	不适用	不适用
CONFIG_BINFMT_MISC	m	m	m	m
CONFIG_BUG	y	y	y	y
CONFIG_BUG_ON_DATA_CORRUPTION	y	y	y	y
CONFIG_CFI_CLANG	不适用	不适用	不适用	不适用
CONFIG_CFI_PERMISSIVE	不适用	不适用	不适用	不适用
CONFIG_COMPAT	y	y	y	y

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_COMPAT_BRK	n	n	n	n
CONFIG_COMPAT_VDSO	不适用	n	不适用	n
CONFIG_DEBUG_CREDENTIALS	n	n	不适用	不适用
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_DEFAULT_MMAP_MIN_ADDR	65536	65536	65536	65536
CONFIG_DEVKMEM	不适用	不适用	不适用	不适用
CONFIG_DEVMEM	n	n	n	n

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_EFI_DISABLE_PCI_DMA	n	n	n	n
CONFIG_FORTIFY_SOURCE	y	y	y	y
CONFIG_HARDENED_USERCOPY	y	y	y	y
CONFIG_HARDENED_USERCOPY_FALLBACK	不适用	不适用	不适用	不适用
CONFIG_HARDENED_USERCOPY_PAGESPAN	不适用	不适用	不适用	不适用
CONFIG_HIBERNATION	y	y	y	y
CONFIG_HW_RANDOM_TPM	不适用	不适用	不适用	不适用
CONFIG_INET_DIAG	m	m	m	m

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_INIT_ON_ALLOC_DEFAULT_ON	n	n	n	n
CONFIG_INIT_ON_FREE_DEFAULT_ON	n	n	n	n
CONFIG_INIT_STACK_ALL_ZERO	不适用	不适用	不适用	不适用
CONFIG_IOMMU_DEFAULT_DMA_STRICT	n	n	n	n
CONFIG_IOMMU_SUPPORT	y	y	y	y
CONFIG_IO_STRICT_DEVMEM	不适用	不适用	不适用	不适用
CONFIG_KEXEC	y	y	y	y
CONFIG_KFENCE	n	n	n	n

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
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_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 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_MODULE_SIG_SHA512	y	y	y	y
CONFIG_PAGE_POISONING	n	n	n	n
CONFIG_PAGE_POISONING_NO_SANITY	不适用	不适用	不适用	不适用
CONFIG_PAGE_POISONING_ZERO	不适用	不适用	不适用	不适用
CONFIG_PANIC_ON_OOPS	y	y	y	y
CONFIG_PANIC_TIMEOUT	0	0	0	0
CONFIG_PROC_KCORE	y	y	y	y
CONFIG_RANDOMIZE_KSTACK_OFFSET_DEFAULT	n	n	n	n

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_RANDOM_TRUST_BOOTLOADER	y	y	不适用	不适用
CONFIG_RANDOM_TRUST_CPU	y	y	不适用	不适用
CONFIG_REFCOUNT_FULL	不适用	不适用	不适用	不适用
CONFIG_SCHED_CORE	不适用	y	不适用	y
CONFIG_SCHED_STACK_END_CHECK	y	y	y	y
CONFIG_SECCOMP	y	y	y	y
CONFIG_SECCOMP_FILTER	y	y	y	y
CONFIG_SECURITY	y	y	y	y
CONFIG_SECURITY_DMESG_RESTRICT	y	y	y	y

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_SECURITY_LANDLOCK	y	y	y	y
CONFIG_SECURITY_LOCKDOWN_LSM	y	y	y	y
CONFIG_SECURITY_LOCKDOWN_LSM_EARLY	y	y	y	y
CONFIG_SECURITY_SELINUX_BOOTPARAM	y	y	y	y
CONFIG_SECURITY_SELINUX_DEVELOP	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	n	n	不适用	不适用
CONFIG_SECURITY_WRITABLE_HOOKS	不适用	不适用	不适用	不适用

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_SECURITY_YAMA	y	y	y	y
CONFIG_SHUFFLE_PAGE_ALLOCATOR	y	y	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y	y	y
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 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_STRICT_DEVMEM	n	n	n	n
CONFIG_STRICT_KERNEL_RWX	y	y	y	y
CONFIG_STRICT_MODULE_RWX	y	y	y	y
CONFIG_SYN_COOKIES	y	y	y	y
CONFIG_VMAP_STACK	y	y	y	y
CONFIG_WERROR	n	n	n	n
CONFIG_ZERO_CALL_USED_REGS	n	n	n	n

允许 ACPI 方法 inserted/replaced 在运行时运行 (CONFI_ACPI_CUSTOM_METHOD)

Amazon Linux 禁用了此选项，因为它允许 `root` 用户写入任意内核内存。

此选项是内核自我保护项目推荐设置之一。

其他二进制格式 (**binfmt_misc**)

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。在中 AL2023，此功能是可选的，是作为内核模块构建的。

BUG() 支持

此选项是内核自我保护项目推荐设置之一。

如果内核在检查内核内存结构的有效性时遇到数据损坏，则执行 BUG()

Linux 内核的某些部分会检查数据结构的内部一致性，并在检测到数据损坏执行 BUG()。

此选项是内核自我保护项目推荐设置之一。

COMPAT_BRK

此选项禁用后（这就是 Amazon Linux 配置内核的方式），则 randomize_va_space sysctl 设置会默认为 2，这也会在 mmap 基础地址、堆栈和 VDSO 页面随机化之上启用随机掩码。

此选项存在于内核中，目的是提供与 1996 年及更早版本的一些古老 libc.so.5 二进制文件的兼容性。

此选项是内核自我保护项目推荐设置之一。

COMPAT_VDSO

此配置选项与 x86-64 有关，与 aarch64 无关。将此设置为 n，Amazon Linux 内核不会在可预测的地址上显示 32 位虚拟动态共享对象 (VDSO)。自 2004 年以来，已知因该选项设置为 n 而被损坏的最新 glibc 版本是 glibc 2.3.3。

此选项是内核自我保护项目推荐设置之一。

CONFIG_DEBUG门控硬化

受限的 Linux 内核配置选项CONFIG_DEBUG通常设计用于为调试问题而构建的内核，性能等问题不是优先事项。 AL2023 启用CONFIG_DEBUG_LIST强化选项。

在配置 IOMMU 之前，禁用 EFI 存根中 PCI 设备的 DMA

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

加强在内核和用户空间之间复制内存的功能

当内核需要将内存复制到用户空间或从用户空间复制内存时，此选项会启用一些检查，可防范某些类别的堆溢出问题。

CONFIG_HARDENED_USERCOPY_FALLBACK 选项存在于内核 4.16 到 5.15 中，可帮助内核开发人员通过 WARN() 发现任何缺失的允许列表条目。由于 AL2023附带了 6.1 内核，因此此选项不再与之相关 AL2023。

该CONFIG_HARDENED_USERCOPY_PAGESPAN选项主要作为开发人员的调试选项存在于内核中 AL2023，不再适用于中的 6.1 内核。

此选项是内核自我保护项目推荐设置之一。

休眠支持

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。需要启用此选项才能支持将休眠按需型实例并支持休眠中断的竞价型实例

随机数生成

AL2023 内核经过配置，确保在 EC2 内有足够的熵可用。

CONFIG_INET_DIAG

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。在中 AL2023，此功能是可选的，是作为内核模块构建的。

在分配和取消分配时将所有内核页和 slab 分配器内存归零

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。 AL2023 由于默认情况下启用此功能可能会影响性能，因此在中禁用了这些选项。CONFIG_INIT_ON_ALLOC_DEFAULT_ON 行为可以通过将 init_on_alloc=1 添加到内核命令行来启用，CONFIG_INIT_ON_FREE_DEFAULT_ON 行为可以通过添加 init_on_free=1 来启用。

将所有堆栈变量初始化为零 (CONFIG_INIT_STACK_ALL_ZERO)

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。此选项需要 GCC 12 或更高，而搭 AL2023 载 GCC 11。

内核模块签名

AL2023 对内核模块的签名进行签名和验证。为了保持构建第三方模块的用户的兼容性，尚未启用要求模块具有有效签名的 CONFIG_MODULE_SIG_FORCE 选项。对于想要确保所有内核模块都经过签名的用户，可以配置锁定 Linux 安全模块 (LSM)以强制执行此操作。

kexec

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。启用此选项是为了便于使用 kdump 功能。

IOMMUSupport

AL2023 启用 IOMMU 支持。默认情况下，CONFIG_IOMMU_DEFAULT_DMA_STRICT 选项未启用，但可以通过将 iommu.passthrough=0 iommu.strict=1 添加到内核命令行来配置此功能。

kfence

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

传统 pty 支持

AL2023 使用现代PTY界面 (devpts)。

此选项是内核自我保护项目推荐设置之一。

锁定 Linux 安全模块 (LSM)

AL2023 构建 lockdown LSM，它将在使用安全启动时自动锁定内核。

CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY 选项未启用。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。不使用安全启动时，可以启用锁定 LSM 并根据需要进行配置。

页面中毒

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。同样在分配和取消分配时将所有内核页和 slab 分配器内存归零，由于这可能会影响性能，因此在 AL2023 内核中禁用了该功能。

堆栈保护器

内 AL2023 核是使用堆栈保护器功能构建的，可通过GCC选项启用。-fstack-protector-strong

此选项是内核自我保护项目推荐设置之一。

seccomp BPFAPI

诸如 systemd 和容器运行时之类的软件使用 seccomp 强化功能来强化用户空间应用程序。

此选项是内核自我保护项目推荐设置之一。

panic() 超时

内 AL2023 核配置时将此值设置为0，这意味着内核在死机后不会重新启动。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。这可以通过 sysctl、/proc/sys/kernel/panic 和在内核命令行上进行配置。

安全模型

AL2023 默认情况下 SELinux ，在允许模式下启用。有关更多信息，请参阅 为 AL2023 设置 SELinux 模式。

锁定 Linux 安全模块 (LSM) 和 yama 模块也已启用。

/proc/kcore

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

在系统调用进入时进行内核栈偏移随机化

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。这可以通过在内核命令行上设置 randomize_kstack_offset=on 来启用。

引用计数检查 (**CONFIG_REFCOUNT_FULL**)

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。由于此选项可能对性能产生影响，因此目前未启用。

调度程序对 SMT 内核的感知 (**CONFIG_SCHED_CORE**)

内 AL2023 核是用它构建的CONFIG_SCHED_CORE，它允许用户空间应用程序使用prctl(PR_SCHED_CORE)。此选项是内核自我保护项目推荐设置之一。

在调用 **schedule()** 时检查是否存在栈损坏 (**CONFIG_SCHED_STACK_END_CHECK**)

内 AL2023 核是在CONFIG_SCHED_STACK_END_CHECK启用状态下构建的。此选项是内核自我保护项目推荐设置之一。

内存分配器强化

内 AL2023 核允许使用CONFIG_SHUFFLE_PAGE_ALLOCATOR、CONFIG_SLAB_FREELIST_HARDENED和选项对内核内存分配器进行强化。CONFIG_SLAB_FREELIST_RANDOM此选项是内核自我保护项目推荐设置之一。

SLUB 调试支持

AL2023 内核之所以CONFIG_SLUB_DEBUG启用，是因为此选项为分配器启用了可选的调试功能，这些功能可以在内核命令行上启用。此选项是内核自我保护项目推荐设置之一。

CONFIG_STATIC_USERMODEHELPER

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。这是因为 CONFIG_STATIC_USERMODEHELPER 需要该发行版的特殊支持，而 Amazon Linux 中目前没有这种支持。

只读内核文本和 rodata（**CONFIG_STRICT_KERNEL_RWX** 和 **CONFIG_STRICT_MODULE_RWX**）

内 AL2023 核配置为将内核和内核模块的文本和rodata内存标记为只读，将非文本内存标记为不可执行。此选项是内核自我保护项目推荐设置之一。

TCP syncookie 支持 (**CONFIG_SYN_COOKIES**)

该 AL2023 内核是在支持 TCP 同步 cookie 的情况下构建的。此选项是内核自我保护项目推荐设置之一。

带有保护页面的虚拟映射栈 (**CONFIG_VMAP_STACK**)

内 AL2023 核是用构建的CONFIG_VMAP_STACK，允许使用保护页面进行虚拟映射的内核堆栈。此选项是内核自我保护项目推荐设置之一。

使用编译器警告作为错误进行构建 (**CONFIG_WERROR**)

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

在函数退出时进行寄存器清零 (**CONFIG_ZERO_CALL_USED_REGS**)

尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

用户空间分配的最小地址

此强化选项可以帮助减少内核 NULL 指针错误的影响。此选项是内核自我保护项目推荐设置之一。

clang特定的强化选项

内 AL2023 核是使用GCC而不是构建的clang，因此无法启用CONFIG_CFI_CLANG强化选项，这也使得它CONFIG_CFI_PERMISSIVE不适用。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

x86-64 特定的内核强化选项

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_AMD_IOMMU	不适用	y	不适用	y
CONFIG_AMD_IOMMU_V2	不适用	y	不适用	不适用
CONFIG_IA32_EMULATION	不适用	y	不适用	y
CONFIG_INTEL_IOMMU	不适用	y	不适用	y
CONFIG_INTEL_IOMMU_DEFAULT_ON	不适用	n	不适用	n
CONFIG_INTEL_IOMMU_SVM	不适用	n	不适用	n

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_LEGACY_VSYSCALL_NONE	不适用	n	不适用	n
CONFIG_MODIFY_LDT_SYSCALL	不适用	n	不适用	n
CONFIG_PAGE_TABLE_ISOLATION	不适用	y	不适用	不适用
CONFIG_RANDOMIZE_MEMORY	不适用	y	不适用	y
CONFIG_X86_64	不适用	y	不适用	y
CONFIG_X86_MSR	不适用	y	不适用	y
CONFIG_X86_VSYSCALL_EMULATION	不适用	y	不适用	y
CONFIG_X86_X32	不适用	不适用	不适用	不适用
CONFIG_X86_X32_ABI	不适用	n	不适用	n

x86-64 支持

基本 x86-64 支持包括物理地址扩展 (PAE) 和不执行 (NX) 位支持。此选项是内核自我保护项目推荐设置之一。

AMD 和英特尔 IOMMU 支持

该 AL2023 内核构建时支持 AMD 和英特尔IOMMUs。此选项是内核自我保护项目推荐设置之一。

CONFIG_INTEL_IOMMU_DEFAULT_ON 选项未设置，但可以通过将 intel_iommu=on 传递到内核命令行来启用。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

中当前未启用该CONFIG_INTEL_IOMMU_SVM选项 AL2023。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

支持 32 位用户空间

Important

对 32 位 x86 用户空间的支持已弃用，在未来的主版本 Amazon Linux 中，可能会取消对运行 32 位用户空间二进制文件的支持。

Note

虽然 AL2023 不再包含任何 32 位软件包，但内核仍将支持运行 32 位用户空间。请参阅32 位 x86 (i686) 软件包了解更多信息。

要支持运行 32 位用户空间应用程序，请不要启用该CONFIG_X86_VSYSCALL_EMULATION选项， AL2023 而是启用CONFIG_IA32_EMULATIONCONFIG_COMPAT、和CONFIG_X86_VSYSCALL_EMULATION选项。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

未启用 64 位处理器的 x32 本机 32 位 ABI (CONFIG_X86_X32 和 CONFIG_X86_X32_ABI) 。此选项是内核自我保护项目推荐设置之一。

x86 型号特定寄存器 (MSR) 支持

启用 CONFIG_X86_MSR 选项是为了支持 turbostat。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

modify_ldt 系统调用

AL2023 不允许用户程序使用 syscall 修改 x86 本地描述符表 (LDT)。modify_ldt此调用是运行 16 位或分段代码所必需的，缺少这个调用可能会破坏 dosemu 等软件，在 WINE 下运行一些程序以及一些非常旧的线程库。此选项是内核自我保护项目推荐设置之一。

在用户模式下移除内核映射

AL2023 配置内核，使大多数内核地址不会映射到用户空间。此选项是内核自我保护项目推荐设置之一。

随机化内核内存部分

AL2023 将内核配置为随机化内核内存部分的基本虚拟地址。此选项是内核自我保护项目推荐设置之一。

aarch64 特定的内核强化选项

CONFIG 选项	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64
CONFIG_ARM64_BTI	y	不适用	y	不适用
CONFIG_ARM64_BTI_KERNEL	不适用	不适用	不适用	不适用
CONFIG_ARM64_PTR_AUTH	y	不适用	y	不适用
CONFIG_ARM64_PTR_AUTH_KERNEL	y	不适用	y	不适用

CONFIG 选项	AL2023/6.1/ aarch64	AL2023/6.1/ x86_64	AL2023/6.12/ aarch64	AL2023/6.12/ x86_64
CONFIG_ARM64_SW_TTBR0_PAN	y	不适用	y	不适用
CONFIG_UNMAP_KERNEL_AT_EL0	y	不适用	y	不适用

分支目标识别

内 AL2023 核支持分支目标识别 (CONFIG_ARM64_BTI)。此选项是内核自我保护项目推荐设置之一。

该CONFIG_ARM64_BTI_KERNEL选项未在 AL2023 构建时启用GCC，并且由于 gcc 错误，上游内核中目前已禁用使用此选项构建内核的支持。尽管此选项是内核自我保护项目 (KSPP) 推荐设置之一，但并 AL2023 未将此配置选项设置为 KSPP 建议的选项。

指针身份验证 (**CONFIG_ARM64_PTR_AUTH**)

该 AL2023 内核支持指针身份验证扩展（ ARMv8.3 扩展的一部分），该扩展可用于帮助缓解面向返回的编程 (ROP) 技术。Graviton 3 引入了在 Graviton 上进行指针身份验证所需的硬件支持。

CONFIG_ARM64_PTR_AUTH 选项已启用，并支持针对用户空间的指针身份验证。由于该CONFIG_ARM64_PTR_AUTH_KERNEL选项也已启用，因此 AL2023 内核可以自行使用返回地址保护。

此选项是内核自我保护项目推荐设置之一。

使用 **TTBR0_EL1** 切换的 Emulate Privileged Access Never

此选项可防止内核直接访问用户空间内存，TTBR0_EL1 只由用户访问例程临时设置为有效值。

此选项是内核自我保护项目推荐设置之一。

在用户空间中运行时取消内核映射

内 AL2023 核配置为在用户空间 (CONFIG_UNMAP_KERNEL_AT_EL0) 中运行时取消内核映射。此选项是内核自我保护项目推荐设置之一。

AL2023 上的 UEFI 安全启动

AL2023 从 2023.1 版本开始支持 UEFI 安全启动。您必须将 AL2023 用于同时支持 UEFI 和 UEFI 安全启动的 Amazon EC2 实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的在 UEFI 启动模式下启动 Amazon EC2 实例的要求。

启用 UEFI 安全启动的 AL2023 实例仅接受由 Amazon 签名的内核级代码（包括 Linux 内核及其模块），从而确保您的实例仅运行由 AWS 签名的内核级代码。

有关 Amazon EC2 实例和 UEFI 安全启动的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 实例的 UEFI 安全启动。

先决条件

- 您必须使用有关 AL2023 2023.1 版本或更高版本的 AMI。
- 实例类型必须支持 UEFI 安全启动。有关更多信息，请参阅《Amazon EC2 用户指南》中的在 UEFI 启动模式下启动 Amazon EC2 实例的要求。

在 AL2023 上启用 UEFI 安全启动

标准 AL2023 AMI 包含由我们的密钥签名的引导加载程序和内核。您可以通过两种方式启用 UEFI 安全启动：一是通过注册现有实例，二是通过注册快照映像创建已预启用 UEFI 安全启动的 AMI。在标准 AL2023 AMI 上，默认不启用 UEFI 安全启动。

AL2023 AMI 的启动模式设置为 `uefi-preferred`，这确保了使用这些 AMI 启动的实例将使用 UEFI 固件（如果实例类型支持 UEFI）。如果实例类型不支持 UEFI，则使用传统 BIOS 固件启动实例。在传统 BIOS 模式下启动实例时，不会强制执行 UEFI 安全启动。

有关 Amazon EC2 实例上 AMI 启动模式的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 启动模式的实例启动行为。

主题

注册现有实例

要注册一个现有实例，请使用一组密钥填充特定的 UEFI 固件变量，这组密钥使固件能够验证引导加载程序，而引导加载程序又能在下次启动时验证内核。

1. Amazon Linux 提供一个简化注册流程的工具。请运行以下命令以使用一组必要的密钥和证书预置实例。

```
sudo amazon-linux-sb enroll
```

2. 运行以下命令来重新启动实例。实例重启后，将启用 UEFI 安全启动。

```
sudo reboot
```

> **Note**
>
> Amazon Linux AMI 目前不支持 Nitro 可信平台模块 (NitroTPM)。如果您除了需要 UEFI 安全启动还需要 NitroTPM ，请参阅下一节的信息。

注册快照映像

当您使用 Amazon EC2 `register-image` API 注册来自 Amazon EBS 根卷快照的 AMI 时，可以使用包含 UEFI 变量存储状态的二进制 blob 来预置 AMI。通过提供 AL2023 `UefiData`，您可以启用 UEFI 安全启动，并且无需按照上一节中的步骤操作。

有关创建和使用二进制 blob 的更多信息，请参阅《Amazon EC2 用户指南》中的创建包含预填充变量存储的二进制 blob。

AL2023 提供一个可以直接在 Amazon EC2 实例上使用的预构建的二进制 blob。该二进制 blob 位于正在运行的实例上的 `/usr/share/amazon-linux-sb-keys/uefi.vars` 中。该 blob 由默认安装在自 2023.1 版本起的 AL2023 AMI 上的 `amazon-linux-sb-keys` RPM 软件包提供。

> **Note**
>
> 为确保您使用的是最新版本的密钥和撤消，请使用来自您创建 AMI 时所使用的同一 AL2023 版本的 blob。

当您注册映像时，建议您使用设置为 uefi 的 RegisterImage API 参数 BootMode。这使得您可以通过将 TpmSupport 参数设置为 v2.0 来启用 NitroTPM。此外，BootMode 设置为 uefi 可以确保启用 UEFI 安全启动，并且在切换到不支持 UEFI 的实例类型时无法意外禁用该模式。

有关 NitroTPM 的更多信息，请参阅《Amazon EC2 用户指南》中的 Amazon EC2 实例的 NitroTPM。

撤消更新

Amazon Linux 可能需要发布使用更新的密钥签名的新版引导加载程序 grub2 或 Linux 内核。在这种情况下，就需要撤消旧密钥，以防止以前版本的引导加载程序中的可利用漏洞绕过 UEFI 安全启动验证流程。

grub2 或 kernel 软件包的更新始终会自动将撤消列表更新到正在运行的实例的 UEFI 变量存储中。这意味着，在 UEFI 安全启动启用的情况下，您在为软件包安装安全更新后，将无法再运行该软件包的旧版本。

UEFI 安全启动在 AL2023 上的工作原理

与其他 Linux 发行版不同，Amazon Linux 不提供一个名为 shim 的额外组件来充当第一阶段引导加载程序。shim 通常使用 Microsoft 密钥签名。例如，在提供 shim 的 Linux 发行版上，shim 会加载 grub2 引导加载程序，该引导加载程序使用 shim 自己的代码来验证 Linux 内核。此外，shim 在机器所有者密钥 (MOK) 数据库中维护自己的一组密钥和撤消，该数据库位于 UEFI 变量存储中并使用 mokutil 工具进行控制。

Amazon Linux 不提供 shim。因为 AMI 所有者控制 UEFI 变量，所以不需要此中间步骤，而此中间步骤会对启动和引导时间产生不利影响。此外，我们默认选择不包括对任何供应商密钥的信任，这减少了执行不想要的二进制文件的机会。当然，如果客户选择信任，则可以包括想要的二进制文件。

使用 Amazon Linux，UEFI 可以直接加载和验证我们的 grub2 引导加载程序。grub2 引导加载程序已修改为在加载 Linux 内核后，使用 UEFI 对其进行验证。因此，使用存储在通常 UEFI db 变量（授权密钥数据库）中的相同证书验证 Linux 内核，并根据与引导加载程序和其他 UEFI 二进制文件相同的 dbx 变量（撤消数据库）测试 Linux 内核。因为我们提供自己的 PK 和 KEK 密钥来控制对 db 数据库和 dbx 数据库的访问，所以我们可以根据需要分发签名的更新和撤消，而无需像 shim 这样的中介。

有关 UEFI 安全启动的更多信息，请参阅《Amazon EC2 用户指南》中的 UEFI 安全启动如何与 Amazon EC2 实例配合工作。

注册您自己的密钥

如上一节所述，对于 Amazon EC2 上的 UEFI 安全启动，Amazon Linux 不需要 shim。当您阅读其他 Linux 发行版的文档时，可能会发现使用 mokutil（在 AL2023 上不存在）来管理机器所有者密钥

(MOK) 数据库的文档。shim 和 MOK 环境解决了 UEFI 固件中对于密钥注册的一些限制，但这些限制不适用于 Amazon EC2 实施 UEFI 安全启动的方式。Amazon EC2 有一套机制可以用来轻松直接地操作 UEFI 变量存储中的密钥。

如果您想注册自己的密钥，可以通过操作现有实例内的变量存储（请参阅从实例内部向变量存储添加密钥）或构建预填充的二进制 blob（请参阅创建包含预填充变量存储的二进制 blob）来实现。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。