



ユーザーガイド

Amazon Linux 2023



Amazon Linux 2023: ユーザーガイド

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

Amazon の商標およびトレードドレスは、お客様に混乱を招く可能性がある態様、または Amazon の信用を傷つけたり、失わせたりする態様において、Amazon のものではない製品またはサービスに関連して使用してはなりません。Amazon が所有していないその他のすべての商標は Amazon との提携、関連、支援関係の有無にかかわらず、それら該当する所有者の資産です。

Table of Contents

Amazon Linux 2023 とは	1
リリース頻度	1
メジャーリリースおよびマイナーリリース	2
新しいリリースの消費	2
長期的なサポートポリシー	2
命名およびバージョニング	3
パフォーマンスと運用の最適化	4
Fedora との関係	5
カスタマイズされた cloud-init	6
セキュリティ更新および機能	7
更新の管理	8
クラウド内のセキュリティ	8
SELinux モード	8
コンプライアンスプログラム	8
SSH サーバーのデフォルト	8
OpenSSL 3 の主な機能	9
ネットワークサービス	9
コアツールチェーンパッケージ glibc、gcc、binutils	10
パッケージ管理ツール	11
デフォルトの SSH サーバー設定	11
廃止された機能	14
compat- パッケージ	14
AL1 で中止され AL2 で削除された廃止済みの機能	14
32 ビット x86 (i686) AMI	15
aws-apitools-* から AWS CLI への切り替え	15
AL2 での upstart から systemd への切り替え	16
AL2 で廃止され、AL2023 で削除された機能	16
32 ビット x86 (i686) パッケージ	17
aws-apitools-* に置き換え AWS CLI	17
amazon-cloudwatch-agent による awslogs の置き換え	18
bzip リビジョン制御システム	18
cgroup v1	18
log4j ホットパッチ (log4j-cve-2021-44228-hotpatch)	19
lsb_release および system-lsb-core パッケージ	19

mcrypt	19
OpenJDK 7 (java-1.7.0-openjdk)	20
Python 2.7	20
rsyslog-openssl による rsyslog-gnutls の置き換え	21
Network Information Service (NIS) / yp	21
Amazon VPC create-dhcp-options 内の複数のドメイン名	21
glibc の Sun RPC	22
audit ログの OpenSSH キーフィンガープリント	22
ld.gold リンカー	22
ping6	22
ftp パッケージ	22
AL2023 で廃止	24
32bit x86 (i686) ランタイムのサポート	25
aspell	25
Berkeley DB (libdb)	25
cron	26
IMDSv1	26
pcre バージョン 1	26
System V init (sysvinit)	27
EOL パッケージが非推奨に	27
AL2 と AL2023 の比較	28
追加、更新、削除されたパッケージ	29
各リリースのサポート	29
命名およびバージョニングの変更	29
最適化	29
複数のアップストリームから供給されています。	30
ネットワークシステムサービス	30
パッケージマネージャー	30
cloud-init の使用	30
グラフィカルデスクトップサポート	31
コンパイラトリプレット	31
32 ビット x86 (i686) パッケージ	31
lsb_release および system-lsb-core パッケージ	32
EPEL	32
axel - HTTP/FTP クライアント	34
brotli および libbrotli - 圧縮	35

collectd - 統計収集デーモン	35
cpulimit	35
exim - メール転送エージェント	35
fuse3 - ユーザースペースのファイルシステム (FUSE) v3	36
ganglia - 分散モニタリングシステム	36
git-lfs - Git を使用した大きなファイルのバージョン管理	36
haveged - HAVEGE アルゴリズムを使用したエントロピーソース	36
inotify-tools - inotify コマンドラインツール	36
iperf - TCP/UDP パフォーマンスベンチマーク	37
jemalloc - malloc に代わる実装	37
libbsd - BSD 互換関数ライブラリ	38
libserf - HTTP クライアントライブラリ	38
libzstd - zstd 圧縮ライブラリ	38
lighttpd ウェブサーバー	38
lshell - 制限付きシェル	38
monit - プロセス、ファイル、ディレクトリ、デバイスモニター	39
nodejs	39
perl-Config-General	40
python2-lockfile - ファイルロック	40
python2-rsa - 純粋な Python RSA	40
python2-simplejson - Python 2 の JSON ルーチン	41
rkhunter - ルートキットハンター	41
rssh - OpenSSH で使用するための制限付きシェル	41
sscg - 自己署名 SSL 証明書ジェネレーター	42
stress - ストレステスト	42
stress-ng - ストレステスト	42
tmpwatch - 最終アクセス時間に基づいてファイルを削除する	42
xmlstarlet - コマンドライン XML ユーティリティ	42
Python 2.7 は Python 3 に置き換えられました	42
セキュリティ更新	43
SELinux	43
OpenSSL 3	44
IMDSv2	44
log4j ホットパッチ (log4j-cve-2021-44228-hotpatch) の削除	45
安定性向上のための確定的な更新	45
デフォルトの Amazon EBS ボリュームとしての gp3	46

統合コントロールグループ階層 (cgroup v2)	46
systemd タイマーによる cron の置き換え	47
ツールチェーンの改善: gcc、binutils、および glibc	47
systemd ジャーナルによる rsyslog の置き換え	47
パッケージの依存関係の最小化	48
curl および libcurl のパッケージ変更	48
GNU プライバシーガード (GNUPG)	49
デフォルト JVM としての Amazon Corretto	49
AWS CLIv2	49
UEFI 優先ブートと Secure Boot	49
SSH サーバーのデフォルト設定の変更	50
AL2023 でのカーネルの変更点 (AL2 との比較)	50
IPv4 TTL	50
セキュリティに重点を置いたカーネル設定の変更	51
その他のカーネル設定の変更	55
カーネルファイルシステムのサポート	57
/tmp 変更	62
AMI とコンテナイメージの変更	62
Amazon Linux 2 と AL2023 AMI の比較	63
Amazon Linux 2 と AL2023 最小 AMI の比較	96
Amazon Linux 2 と AL2023 コンテナの比較	117
AL1 と AL2023 の比較	125
各リリースのサポート	125
init システムとして systemd が upstart を置き換えます。	126
Python 2.6 および 2.7 は Python 3 に置き換えられました	126
最も古い JDK としての OpenJDK 8	126
AL2023 でのカーネルの変更点 (AL1 との比較)	126
カーネルライブパッチ	126
カーネルファイルシステムのサポート	126
セキュリティ重視のカーネル設定の変更	129
その他のカーネル設定の変更	132
AL1 と AL2023 AMI の比較	133
AL1 と AL2023 の最小 AMI の比較	167
AL1 と AL2023 のコンテナの比較	187
システム要件	196
AL2023 を実行するための CPU 要件	196

AL2023 の ARM CPU 要件	196
AL2023 の x86-64 CPU 要件	197
AL2023 を実行するためのメモリ (RAM) 要件	198
Amazon Linux カーネル	199
カーネルライフサイクル	199
カーネルの更新	200
ディストリビューション間のカーネルバージョンの重複	200
CVE の処理	201
実行すべきこと	201
NVIDIA ドライバー	202
NVIDIA リポジトリについて	202
NVIDIA リポジトリの有効化	202
NVIDIA ドライバーのインストール	203
CUDA ツールキットのインストール	203
NVIDIA リポジトリの削除	204
グラフィカルデスクトップ	205
関連トピック	205
Amazon Linux の補足パッケージ	206
Amazon Linux (または SPAL) のサプリメンタルパッケージとは	206
利点	206
SPAL パッケージのサポート	207
パッケージ関連の問題のレポート	207
関連トピック	208
AL2023 で SPAL を設定する	208
前提条件	209
前提条件の確認	209
システムに SPAL をインストールする	209
SPAL パッケージのインストール	211
SPAL ソースパッケージのダウンロード	211
SPAL debuginfo パッケージのインストール	211
システムから SPAL リポジトリをアンインストールする	212
関連トピック	212
アプリケーションの実行	213
systemd によるリソースコントロール	213
1 回限りのコマンドを実行するための systemd-run によるリソースコントロール	213
systemd サービスでのリソースコントロール	216

cggroups ユーティリティの使用	221
での AL2023 の使用 AWS	223
の開始方法 AWS	223
にサインアップする AWS アカウント	223
プログラムによるアクセス権を付与する	223
Amazon EC2 での AL2023	225
Amazon EC2 コンソールを使用した AL2023 の起動	226
SSM パラメータと を使用して AL2023 を起動する AWS CLI	227
を使用して最新の AL2023 AMI を起動する CloudFormation	228
特定の AMI ID を使用して AL2023 を起動する	230
AL2023 AMI の非推奨とライフサイクル	230
AL2023 インスタンスへの接続	231
AL2023 標準 (デフォルト) と最小 AMI の比較	231
コンテナでの AL2023	259
AL2023 ベースコンテナイメージ	260
AL2023 最小コンテナイメージ	262
必要最低限の AL2023 コンテナイメージの構築	264
AL2023 コンテナイメージパッケージリストの比較	268
AL2023 コンテナイメージと最小 AMI の比較	274
Elastic Beanstalk での AL2023	291
AL2023 CloudShell	291
Amazon ECS コンテナホスト用 AL2023	292
Amazon ECS 関連の変更 (AL2 以降)	292
Amazon ECS 最適化 AMI をカスタマイズする	293
AL2023 での Amazon EFS	293
amazon-efs-utils	294
Amazon EFS ファイルシステムのマウンティング	294
AL2023 での Amazon EMR	295
AL2023 ベースの Amazon EMR リリース	295
EKS での Amazon EMR	295
での AL2023 AWS Lambda	295
provided.al2023 Lambda ランタイム	295
AL2023 ベースのランタイム	296
チュートリアル	297
AL2023 への LAMP のインストール	297
ステップ 1: LAMP サーバーを準備する	298

ステップ 2: LAMP サーバーをテストする	303
ステップ 3: データベースサーバーをセキュリティで保護する	305
ステップ 4: (オプション) phpMyAdmin をインストールする	306
トラブルシューティング	309
関連トピック	310
AL2023 に SSL/TLS を設定する	310
前提条件	312
ステップ 1: サーバーで TLS を有効にする	313
ステップ 2: CA 署名証明書を取得する	316
ステップ 3: セキュリティ設定をテストして強化する	324
トラブルシューティング	327
AL2023 で WordPress ブログをホストする	328
前提条件	329
WordPress のインストール	330
次のステップ	340
ヘルプ! パブリック DNS 名が変更されたため、ブログが壊れました	342
AL2023 での Redis 6 から Valkey への移行	343
Redis 6 のサポートタイムライン	343
Valkey の概要	343
移行の計画とタイムライン	343
移行のオプションとステップ	344
関連トピック	347
AL2023 に GNOME をインストールする	347
前提条件	348
インストール	348
関連トピック	348
AL2023 で VNC を設定する	349
前提条件	349
ステップ 1: インストール	349
ステップ 2: 設定	350
ステップ 3: VNC クライアントを使用して接続する	351
(オプション) 起動時にサービスを開始する	352
(オプション) アイドルロック画面を無効にする	353
関連トピック	353
AL2023 カーネルでの多世代 LRU (MGLRU) の使用	353
設定と調整	354

AL2023 に LibreOffice をインストールする	355
前提条件	355
ステップ 1: システムアーキテクチャを決定する	356
ステップ 2: LibreOffice RPM パッケージをダウンロードする	356
ステップ 3: アーカイブを抽出する	357
ステップ 4: RPM パッケージをインストールする	357
ステップ 5: インストールを確認する	358
LibreOffice の起動	358
トラブルシューティング	359
Amazon EC2 以外の AL2023	360
AL2023 VM イメージをダウンロードする	360
サポートされる設定	360
KVM の要件	361
VMware の要件	363
Hyper-V の要件	365
AL2023 VM 設定	367
NoCloud seed.iso ベースの設定	368
VMware guestinfo ベースの設定	372
標準 AMI と KVM イメージ間での AL2023 パッケージリストの比較	374
標準 AMI と VMware OVA イメージ間での AL2023 パッケージリストの比較	399
標準 AMI と Hyper-V イメージ間での AL2023 パッケージリストの比較	424
Amazon Linux のバージョンの識別	450
/etc/os-release	450
主な違い	451
フィールドタイプ	451
/etc/os-release の例	453
他のディストリビューションとの比較	454
Amazon Linux 固有	456
/etc/sytem-release	457
/etc/image-id	457
Amazon Linux 固有の例	458
サンプルのコード	460
ファイルシステムのレイアウト	473
/	473
/boot	474
/boot/efi	474

/etc	474
/home	474
/root	475
/srv	475
/tmp	476
/run	477
/usr	477
/usr/bin	478
/usr/include	478
/usr/lib および /usr/lib64	478
/usr/local	478
/usr/share	478
/var	479
/var/cache	479
/var/lib	479
/var/log	479
/var/spool	480
/var/tmp	480
AL2023 の更新	481
更新を安全にデプロイするためのベストプラクティス	481
マイナーな更新のための準備	484
メジャーな更新の準備	484
新しいアップデートの通知を受け取る	485
バージョン管理されたリポジトリを介した確定的なアップグレード	486
メジャーリリースとマイナーリリースから受け取る更新を管理する	486
メジャーバージョンとマイナーバージョンのアップグレードの違い	487
更新が利用可能になる時期の確認	487
AL2023 リポジトリから入手できるパッケージ更新を管理する	488
インスタンスの置き換え	488
確定的なインプレースアップグレード	489
更新の管理	497
使用可能なパッケージ更新の確認	498
DNF およびリポジトリバージョンを使用してセキュリティ更新を適用します。	502
(セキュリティ) 更新後の自動サービス再起動	515
セキュリティ更新を適用する際、いつ再起動が必要になりますか?	516
最新のリポジトリバージョンを有効にしたインスタンスの起動	517

パッケージサポート情報の取得	517
dnf check-release-update	518
新しいリポジトリの追加、有効化、無効化	522
cloud-init によるリポジトリの追加	525
カーネルライブパッチ	526
制限事項	527
サポートされている構成と前提条件	527
カーネルライブパッチを使用する	528
カーネルの更新	533
AL2023 での Linux カーネルバージョン	533
AL2023 を新しいカーネルバージョンに更新する	534
AL2023 カーネル - よくある質問	539
プログラミング言語とランタイム	541
C/C++ と Fortran	541
GCC 14	542
言語標準バージョンの比較	543
.NET	545
新しい .NET バージョンへの移行	545
.NET のインストール	545
Go	546
AL2023 Lambda 関数: Go	546
Java	546
Node.js	39
セキュリティのベストプラクティス	548
Perl	548
Perl モジュール	549
PHP	549
新しい PHP バージョンへの移行	549
PHP 7.x からの移行	550
PHP モジュール	550
Python	550
新しい Python バージョンへの移行	551
Python モジュール	551
Ruby	551
Rust	553
AL2023 Lambda 関数: Rust	554

Swift	554
AL2023 関数: Swift	555
TypeScript	555
AL2023 予約済みユーザーおよびグループ	560
AL2023 の予約済みユーザーのリスト	560
AL2023 リザーブドグループのリスト	568
AL2023 で利用可能なコーデック	581
セキュリティおよびコンプライアンス	584
セキュリティアドバイザリ	585
ALAS Announcements	585
ALAS のよくある質問	586
ALAS アドバイザリ	586
アドバイザリと RPM リポジトリ	586
Advisory ID	587
アドバイザリのリリース日とアドバイザリの更新日	587
アドバイザリのタイプ	588
アドバイザリの重大度	588
アドバイザリとパッケージ	589
アドバイザリと CVE	590
アドバイザリのテキスト	590
カーネルライブパッチのアドバイザリ	591
updateinfo.xml スキーマ	592
適用可能なアドバイザリの一覧表示	592
インプレース更新	596
アドバイザリに記載されている更新プログラムの適用	597
AL2023 の SELinux モードの設定	600
AL2023 でのデフォルトの SELinux のステータスとモード	601
enforcing モードへの変更	601
SELinux を無効にするオプション	603
AL2023 で FIPS モードを有効にする	604
AL2023 コンテナで FIPS モードを有効にする	606
AL2023 で OpenSSL FIPS プロバイダーを切り替える	608
カーネル強化	610
カーネル強化オプション (アーキテクチャに依存していません)	610
x86-64 固有のカーネル強化オプション	627
aarch64 固有のカーネル強化オプション	630

リポジトリメタデータ署名	632
リポジトリメタデータ署名の仕組み	632
gpgcheck と の違い repo_gpgcheck	633
リポジトリメタデータ検証の有効化	633
リポジトリメタデータ署名が機能していることの確認	634
AL2023 リポジトリの GPG パブリックキー	635
AL2023 での UEFI セキュアブート	635
AL2023 で UEFI セキュアブートを有効にする	636
既存のインスタンスの登録	637
スナップショットからイメージを登録する	637
失効の更新	638
AL2023 での UEFI 安全ブートのしくみ	638
独自のキーを登録する	639
.....	dcxl

Amazon Linux 2023 とは

Amazon Linux 2023 (AL2023) は、Amazon Web Services (AWS) の次世代の Amazon Linux です。AL2023 を使うことで、安全で安定した高性能なランタイム環境で、クラウドおよびエンタープライズアプリケーションを開発、実行できます。また、Linux での最新のイノベーションへのアクセスを含む、長期的なサポートを提供するアプリケーション環境を取得できます。AL2023 は追加料金なしで提供されます。

AL2023 は Amazon Linux 2 (AL2) の後継です。AL2023 と AL2 の違いについては、「[AL2 と AL2023 の比較](#)」および「[AL2023 のパッケージ変更](#)」を参照してください。

トピック

- [リリース頻度](#)
- [命名およびバージョニング](#)
- [パフォーマンスと運用の最適化](#)
- [Fedora との関係](#)
- [カスタマイズされた cloud-init](#)
- [セキュリティ更新および機能](#)
- [ネットワークサービス](#)
- [コアツールチェーンパッケージglibc、gcc、binutils](#)
- [パッケージ管理ツール](#)
- [デフォルトの SSH サーバー設定](#)

リリース頻度

Amazon Linux 2023 (AL2023) は 2023 年 3 月にリリースされ、2029 年 6 月 30 日までサポートされます。サポートには 2 つのフェーズがあります。

- 標準サポートフェーズ - このフェーズでは、四半期ごとにリリースのマイナーバージョン更新が行われます。標準サポートフェーズは 2027 年 6 月 30 日に終了します。
- メンテナンス - このフェーズでは、リリースのセキュリティ更新と重要なバグ修正のみが行われます。これらの更新は、利用可能になるとすぐに公開されます。メンテナンスフェーズは 2029 年 6 月 30 日に終了します。

メジャーリリースおよびマイナーリリース

Amazon Linux のリリース (メジャーバージョン、マイナーバージョン、またはセキュリティリリース) ごとに、新しい Linux Amazon マシンイメージ (AMI) がリリースされます。

- メジャーバージョンリリース — スタック全体のセキュリティとパフォーマンスの新機能と改善が含まれています。改善には、カーネル、ツールチェーン Glib C、OpenSSL、およびその他のシステムライブラリとユーティリティの大幅な変更が含まれる場合があります。Amazon Linux のメジャーリリースの一部は、アップストリームの Fedora Linux 配布の最新バージョンに基づいています。AWS は Fedora 以外の他のアップストリームの特定のパッケージを追加したり、置き換えたりする場合があります。
- マイナーバージョンリリース — セキュリティ更新、バグ修正、および新機能およびパッケージが含まれる四半期ごとの更新です。各マイナーバージョンは、新しい機能やパッケージに加えて、セキュリティやバグの修正を含む更新の累積リストです。これらのリリースには、PHP などの最新の言語ランタイムが含まれる場合があります。Ansible や Docker など、他の一般的なソフトウェアパッケージも含まれる場合があります。

新しいリリースの消費

更新は、新しい Amazon マシンイメージ (AMI) リリースと対応する新しいリポジトリを組み合わせで提供されます。デフォルトでは、新しい AMI とそれが指すリポジトリが結合されます。ただし、実行中の Amazon EC2 インスタンスに更新を適用するために、時間の経過とともに新しいリポジトリバージョンを指定できます。最新の AMI の新しいインスタンスを起動して更新することもできます。

長期的なサポートポリシー

Amazon Linux では、すべてのパッケージの更新が提供され、Amazon Linux 上に構築されたアプリケーションのメジャーバージョン内での互換性が維持されます。glibc ライブラリ、OpenSSL、OpenSSH、および DNF パッケージマネージャなどのコアパッケージは、AL2023 のメジャーリリースの有効期間中サポートを受けられます。コアパッケージに含まれていないパッケージは、特定のアップストリームソースに基づいてサポートされます。以下のコマンドを実行すると、個々のパッケージの具体的なサポート状況および日付を確認できます。

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


以下のコマンドを実行すると、現在インストールされているすべてのパッケージに関する情報を取得できます。

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

コアパッケージのすべてのリストはプレビュー中に完成します。コアパッケージとして含まれるパッケージをさらに確認する場合はお問い合わせください。私たちは、フィードバックを収集して評価しています。AL2023 に関するフィードバックは、指定の AWS 担当者を通じて提供するか、GitHub の[amazon-linux-2023リポジトリ](#)に問題を提出することで提供できます。

命名およびバージョンニング

AL2023 は、2 年間の標準サポート期間中、3 か月ごとにマイナーリリースを提供します。各リリースは 0 から N までの数字で識別されます。0 はそのイテレーションの元のメジャーリリースを指します。すべてのリリースは Amazon Linux 2023 という名前になります。次のバージョンの Amazon Linux がリリースされると、AL2023 は延長サポートに入り、セキュリティ更新と重大なバグ修正の更新が提供されます。

例えば、AL2023 のマイナーリリースのフォーマットは以下のとおりです。

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

対応する AL2023 AMI のフォーマットは以下のとおりです。

- 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

特定のマイナーバージョン内では、AMI リリース日のタイムスタンプが付いた通常の AMI リリースが行われます。

- 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

AL2 または AL2023 インスタンスを識別するための推奨方法は、まず `/etc/system-release-cpe` から共通プラットフォーム一覧 (CPE) 文字列を読み取ることです。次に、文字列をフィールドに分割します。最後に、プラットフォームとバージョンの値を読み取ります。

AL2023 では、プラットフォーム識別用の新しいファイルも導入されています。

- `/etc/system-release` への `/etc/amazon-linux-release` シンボリックリンク
- `/etc/amazon-linux-release-cpe` への `/etc/system-release-cpe` シンボリックリンク

この 2 つのファイルは、インスタンスが Amazon Linux であることを示しています。特定のプラットフォームとバージョンの値を知りたい場合を除いて、ファイルを読み取ったり、文字列をフィールドに分割したりする必要はありません。

パフォーマンスと運用の最適化

Amazon Linux 6.1 カーネル

- AL2023 は、Elastic Network Adapter (ENA) および Elastic Fabric Adapter (EFA) デバイス用の最新のドライバーを使用しています。AL2023 は、Amazon EC2 インフラストラクチャのハードウェアのパフォーマンスと機能のバックポートに重点を置いています。
- x86_64 および aarch64 インスタンスタイプにはカーネルライブパッチが適用されています。これにより、頻繁に再起動する必要が少なくなります。
- カーネルビルドとランタイムのすべての設定には、AL2 と同じパフォーマンス上および運用上の最適化が多数含まれています。

基本ツールチェーンの選択とデフォルトのビルドフラグ

- AL2023 パッケージは、コンパイラ最適化 (`-O2`) がデフォルトで有効になっている状態でビルドされます。
- AL2023 パッケージは x86-64 システム (`-march=x86-64-v2`) の x86-64v2 が必要で、aarch64 (`-march=armv8.2-a+crypto -mtune=neoverse-n1`) の Graviton 2 以降が必要です。
- AL2023 パッケージは、自動ベクトル化を有効にして構築されています (`-ftree-vectorize`)。
- AL2023 パッケージはリンクタイム最適化 (LTO) を有効にしてビルドされています。
- AL2023 は Rust、Clang/LLVM、および Go の更新バージョンを使用しています。

パッケージの選択とバージョン

- 主要なシステムコンポーネントへの一部のバックポートには、Amazon EC2 インフラストラクチャ、特に Graviton インスタンスで実行するためのいくつかのパフォーマンスの向上が含まれています。
- AL2023 には複数の AWS のサービス および機能が統合されています。これには AWS CLI、SSM エージェント、Amazon Kinesis エージェント、および CloudFormation が含まれます。
- AL2023 では、Java 開発キット (JDK) として Amazon Corretto が使用されます。
- AL2023 は、上流プロジェクトによってリリースされる新しいバージョンへのデータベースエンジンとプログラミング言語のランタイム更新を提供します。新しいバージョンのプログラミング言語ランタイムは、リリース時に追加されます。

クラウド環境へのデプロイ

- ベースの AL2023 AMI とコンテナイメージは、パッチ適用インスタンスの置換をサポートするように頻繁に更新されます。
- カーネルの更新は AL2023 AMI の更新に含まれています。つまり、カーネルを更新するために `yum update` や `reboot` などのコマンドを使用する必要がないということです。
- 標準の AL2023 AMI に加えて、最小 AMI とコンテナイメージも利用できます。サービスの実行に必要な最小限のパッケージ数で環境を実行するには、最小 AMI を選択します。
- デフォルトでは、AL2023 AMI とコンテナは特定のバージョンのパッケージリポジトリにロックされます。起動時の自動更新はありません。つまり、パッケージ更新をいつ取り込むかを常に制御できるということです。本番環境にロールアウトする前に、いつでもベータ/ガンマ環境でテストできます。問題が発生した場合は、事前に検証されたロールバックパスを使用できます。

Fedora との関係

AL2023 は、Fedora から独立した独自のリリースおよびサポートライフサイクルを維持しています。AL2023 では、オープンソースソフトウェアの更新バージョン、多種多様なパッケージ、頻繁なリリースが提供されています。これにより、使い慣れた RPM ベースのオペレーティングシステムが維持されます。

AL2023 の一般提供 (GA) バージョンは、特定の Fedora リリースと直接比較できるものではありません。AL2023 GA バージョンには Fedora 34、35、36 のコンポーネントが含まれています。一部のコンポーネントは Fedora のコンポーネントと同じで、一部は変更されています。他のコンポーネントは、CentOS Stream 9 のコンポーネントによく似ているか、独自に開発されたものです。Amazon

Linux カーネルは、Fedora とは別に選択された kernel.org にある長期サポートオプションから供給されています。

カスタマイズされた cloud-init

cloud-init パッケージは、クラウドコンピューティング環境で Linux イメージをブートストラップするオープンソースアプリケーションです。詳細については [cloud-init のドキュメント](#) を参照してください。

AL2023 には、cloud-init のカスタマイズバージョンが含まれています。cloud-init を使用すると、起動時のインスタンスの動作を指定できます。

インスタンスを起動する際、ユーザーデータフィールドを使用して cloud-init にアクションを渡すことができます。つまり、さまざまなユースケースに対して共通の Amazon マシンイメージ (AMI) を使用し、インスタンスの起動時にその AMI を動的に設定できます。AL2023 は ec2-user アカウントの設定にも cloud-init を使用します。

AL2023 は /etc/cloud/cloud.cfg.d と /etc/cloud/cloud.cfg にある cloud-init アクションを使用します。独自の cloud-init アクションファイルを /etc/cloud/cloud.cfg.d ディレクトリに作成できます。Cloud-init はこのディレクトリ内のすべてのファイルを辞書順に読み込みます。後のファイルは以前のファイルの値を上書きします。cloud-init がインスタンスを起動すると、cloud-init パッケージは以下の設定タスクを実行します。

- デフォルトのロケールの設定
- ホスト名の設定
- ユーザーデータの解析および処理
- ホストプライベート SSH キーの生成
- 容易にログインおよび管理できるように、ユーザーのパブリック SSH キーを .ssh/authorized_keys に追加
- パッケージ管理のためのリポジトリの準備
- user-data で定義されたパッケージアクションの処理
- user-data に含まれるユーザースクリプトの実行
- インスタンスストアボリュームのマウント (該当する場合)
 - デフォルトでは、ephemeral0 インスタンスストアボリュームが存在し有効なファイルシステムを含む場合、インスタンスストアボリュームは /media/ephemeral0 にマウントされます。それ以外の場合は、マウントされません。

- デフォルトでは、m1.small および c1.medium インスタンスタイプの場合、インスタンスに関連付けられたすべてのスワップボリュームがマウントされます。
- 以下の cloud-init ディレクティブを使用して、デフォルトのインスタンスストアボリュームマウントを上書きできます。

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

マウントをより制御するには、「cloud-init ドキュメント」の「[マウント](#)」を参照してください。

- インスタンスが起動しても、TRIM をサポートするインスタンスストアボリュームはフォーマット化されません。インスタンスストアボリュームをマウントする前に、インスタンスストアボリュームを分割してフォーマット化する必要があります。

詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスストアボリュームの TRIM のサポート](#)」を参照してください。

- インスタンスを起動する際、disk_setup モジュールを使用してインスタンスストアボリュームを分割およびフォーマット化することができます。

詳細については、「cloud-init ドキュメント」の「[ディスクの設定](#)」を参照してください。

SELinux で cloud-init を使用方法については、「[cloud-init を使用して enforcing モードを有効にする](#)」を参照してください。

cloud-init ユーザーデータフォーマットについては、「cloud-init ドキュメント」の「[User-Data フォーマット](#)」を参照してください。

セキュリティ更新および機能

AL2023 には、多数のセキュリティ更新およびソリューションが用意されています。

トピック

- [更新の管理](#)
- [クラウド内のセキュリティ](#)
- [SELinux モード](#)

- [コンプライアンスプログラム](#)
- [SSH サーバーのデフォルト](#)
- [OpenSSL 3 の主な機能](#)

更新の管理

DNF とリポジトリバージョンを使用してセキュリティ更新を適用します。詳細については、「[AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)」を参照してください。

クラウド内のセキュリティ

セキュリティは、AWS とユーザーの間の共有責任です。[責任共有モデル](#)では、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。詳細については、「[Amazon Linux 2023 でのセキュリティおよびコンプライアンス](#)」を参照してください。

SELinux モード

AL2023 では SELinux はデフォルトで有効になっており、許可モードに設定されています。許可モードでは、アクセス拒否は記録されますが、強制ではありません。

SELinux ポリシーは、ユーザー、プロセス、プログラム、ファイル、デバイスのアクセス許可を定義します。SELinux では、2 つのポリシーから 1 つを選択できます。ポリシーはターゲットを絞ったセキュリティ、またはマルチレベルセキュリティ (MLS) です。

SELinux のモードとポリシーの詳細については、「[AL2023 の SELinux モードの設定 および SELinux プロジェクト Wiki](#)」を参照してください。

コンプライアンスプログラム

独立した監査人が、AL2023 のセキュリティとコンプライアンスを多数の AWS コンプライアンスプログラムとともに評価します。

SSH サーバーのデフォルト

AL2023 には OpenSSH 8.7 が含まれています。デフォルトでは、OpenSSH 8.7 は ssh-rsa キー交換アルゴリズムを無効にします。詳細については、「[デフォルトの SSH サーバー設定](#)」を参照してください。

OpenSSL 3 の主な機能

- 証明書管理プロトコル (CMP、RFC 4210) には CRMF (RFC 4211) と HTTP 転送 (RFC 6712) の両方が含まれています。
- libcrypto の HTTP または HTTPS クライアントは、GET および POST アクション、リダイレクト、プレーンおよび ASN.1 にエンコードされたコンテンツ、プロキシ、タイムアウトをサポートします。
- EVP_KDF はキー派生関数と連携します。
- EVP_MAC API は MACs と連携します。
- Linux カーネル TLS サポート。

詳細については、「[OpenSSL の移行ガイド](#)」を参照してください。

ネットワークサービス

オープンソースプロジェクト systemd-networkd は、最新の Linux 配布で広く使用できます。このプロジェクトでは、systemd フレームワークの他の部分と同様の宣言型設定言語を使用しています。主な設定ファイルタイプは .network および .link ファイルです。

amazon-ec2-net-utils パッケージはインターフェイス固有の設定を /run/systemd/network ディレクトリに生成します。これらの設定では、インターフェイスがインスタンスにアタッチされている場合、IPv4 と IPv6 の両方のネットワークが有効になります。これらの構成では、ローカルソースのトラフィックが対応するインスタンスのネットワークインターフェースを介してネットワークにルーティングされるようにするポリシールーティングルールもインストールされます。これらのルールにより、関連するアドレスまたはプレフィックスから適切なトラフィックが Elastic Network Interface (ENI) 経由でルーティングされるようになります。ENI の使用の詳細については「Amazon EC2 ユーザーガイド」の「[Elastic Network Interface](#)」を参照してください。

このネットワーク動作は、/etc/systemd/network ディレクトリにカスタム設定ファイルを配置して、/run/systemd/network に含まれるデフォルトの設定を上書きすることでカスタマイズできます。

[systemd.network](#) のドキュメントには、systemd-networkd サービスが特定のインターフェースに適用される設定を決定する方法が説明されています。また、ENI ベースのインターフェースには、altnames と呼ばれる、さまざまな AWS リソースのプロパティを反映した代替名も生成

されます。ENI ベースのインターフェースプロパティは ENI アタッチメントの ENI ID および DeviceIndex フィールドです。これらのインターフェースは、ip コマンドなどのさまざまなツールの使用時に、そのプロパティを使用して参照できます。

AL2023 インスタンスのインターフェイス名が、systemd スロット命名スキームを使用して生成されます。詳細については、「[systemd.net 命名スキーム](#)」を参照してください。

さらに、AL2023 は fq_code1 アクティブキュー管理ネットワークの送信スケジューリングアルゴリズムをデフォルトで使用します。詳細については、「[CoDel の概要](#)」を参照してください。

コアツールチェーンパッケージ glibc、gcc、binutils

Amazon Linux のパッケージのサブセットはコアツールチェーンパッケージとして指定されています。AL2023 の主な構成要素として、コアパッケージには 5 年間のサポートが付属します。パッケージのバージョンは変更される場合がありますが、Amazon Linux リリースに含まれるパッケージには長期サポートが適用されます。

これら 3 つのコアパッケージは、Amazon Linux 配布のほとんどのソフトウェアの構築に使用されるシステムツールチェーンを提供します。

パッケージ	定義	目的
glibc 2.34	システム C ライブラリ	標準機能を提供するほとんどのバイナリプログラムや、プログラムとカーネル間のインターフェースで使用されます。
gcc 11.2	gcc コンパイラスイート	コンパイル C、C++、Fortran。
binutils 2.35	アセンブラとリンカー、その他のバイナリツール	バイナリプログラムを操作または検査します。

glibc ライブラリを更新した後も再起動が必要です。サービスを制御する更新の場合は、サービスの再起動のみで更新を取得できる場合があります。しかし、システムを再起動すると、パッケージとライブラリの以前の更新はすべて完了します。

パッケージ管理ツール

AL2023 のデフォルトのソフトウェアパッケージ管理ツールは DNF です。DNF は AL2 のパッケージ管理ツールである YUM の後継です。

DNF の使い方は YUM と似ています。DNF コマンドとコマンドオプションの多くは YUM コマンドと同じです。コマンドラインインターフェイス (CLI) のコマンドでは、ほとんどの場合、dnf が yum に置き換えられます。

例えば、次の AL2 yum コマンドの場合

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

AL2023 では、以下のコマンドになります。

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

AL2023 では、yum コマンドは引き続き使用できますが、dnf コマンドへのポインタとして利用できます。そのため、yum コマンドをシェルまたはスクリプトで使用すると、すべてのコマンドとオプションは DNF CLI と同じになります。YUM CLI と DNF CLI の相違点の詳細については、「[YUM との比較における DNF CLI の変更点](#)」を参照してください。

コマンドと dnf コマンドのオプションの詳細については、マニュアルページ「man dnf」を参照してください。詳細については、「[DNF コマンドリファレンス](#)」を参照してください。

デフォルトの SSH サーバー設定

数年前の SSH クライアントを使用している場合、インスタンスに接続するとエラーが表示される可能性があります。一致するホストキータイプが見つからないというエラーが表示された場合は、SSH ホストキーを更新してこの問題を解決します。

デフォルトの **ssh-rsa** 署名の無効化

AL2023 には、従来の ssh-rsa ホストキーアルゴリズムを無効にして生成するホストキーの数を減らすデフォルト設定が含まれています。クライアントは ssh-ed25519 または ecdsa-sha2-nistp256 ホストキーアルゴリズムをサポートする必要があります。

デフォルト設定では以下のキー交換アルゴリズムを受け入れます。

- 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

デフォルトでは、AL2023 は ed25519 および ECDSA ホストキーを生成します。クライアントは ssh-ed25519 または ecdsa-sha2-nistp256 ホストキーアルゴリズムのいずれかをサポートします。SSH でインスタンスに接続する場合、ssh-ed25519 または ecdsa-sha2-nistp256 などの互換性のあるアルゴリズムをサポートするクライアントを使用する必要があります。他の種類のキーを使用する必要がある場合は、生成されたキーのリストを user-data 内の cloud-config フラグメントで上書きします。

以下の例では、cloud-config は ecdsa および ed25519 キーを使用して rsa ホストキーを生成します。

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

公開キー認証に RSA キーペアを使用する場合、SSH クライアントは rsa-sha2-256 または rsa-sha2-512 署名をサポートしている必要があります。互換性のないクライアントを使用していてアップグレードできない場合は、インスタンスの ssh-rsa サポートを再度有効にします。ssh-rsa サポートを再度有効にするには、次のコマンドを使用して LEGACY システム暗号化ポリシーをアクティブ化します。

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

ホストキーの管理の詳細については、Amazon Linux の「[Host Keys](#)」を参照してください。

AL2023 の廃止された機能

ここでは、AL2 で廃止され、AL2023 に含まれていない機能について説明します。つまり、AL2 には含まれているが、AL2023 には含まれておらず、今後も AL2023 に追加されることがない機能やパッケージのことです。AL2 で機能がサポートされる期間については、「[AL2 の廃止された機能](#)」を参照してください。

AL2023 で廃止され、今後のリリースで削除される機能もあります。この章では、廃止された機能、機能のサポートが終了する時期、Amazon Linux から削除される時期について説明します。廃止された機能を把握しておく、AL2023 をデプロイして Amazon Linux の次のメジャーバージョンに備えるうえで役立ちます。

トピック

- [compat- パッケージ](#)
- [AL1 で中止され AL2 で削除された廃止済みの機能](#)
- [AL2 で廃止され、AL2023 で削除された機能](#)
- [AL2023 で廃止](#)

compat- パッケージ

プレフィックスが compat- の AL2 のパッケージはすべて、パッケージの最新バージョン向けに再構築されていない古いバイナリとのバイナリ互換性を確保するために提供されています。Amazon Linux の新しい各メジャーバージョンでは、以前のリリースの compat- パッケージは一切引き継がれません。

Amazon Linux (AL2 など) のリリースの compat- パッケージはすべて廃止され、後続のバージョン (AL2023 など) には含まれていません。ライブラリの更新バージョンに対してソフトウェアを再構築することを強くお勧めします。

AL1 で中止され AL2 で削除された廃止済みの機能

このセクションでは、AL1 で使用できたものの AL2 では使用できなくなった機能について説明します。

Note

AL1 のメンテナンスサポート段階の一環として、一部のパッケージには AL1 のサポート終了日より早いサポート終了日が設定されています。詳細については、「[AL1 パッケージのサポートステートメント](#)」を参照してください。

Note

一部の AL1 機能は、以前のリリースで廃止されています。詳細については、「[AL1 リリースノート](#)」を参照してください。

トピック

- [32 ビット x86 \(i686\) AMI](#)
- [aws-apitools-* から AWS CLI への切り替え](#)
- [AL2 での upstart から systemd への切り替え](#)

32 ビット x86 (i686) AMI

[AL1 の 2014.09 リリース](#)において、これが 32 ビット AMI を提供する最後のリリースになることが発表されました。したがって、[AL1 の 2015.03 リリース](#)以降、Amazon Linux では 32 ビットモードでのシステム実行をサポートしていません。AL2 では、x86-64 ホスト上の 32 ビットバイナリのランタイムサポートを限定的に提供しており、新しい 32 ビットバイナリの構築を可能にする開発パッケージは提供していません。AL2023 には 32 ビットユーザースペースパッケージは含まれなくなりました。AL2023 に移行する前に、64 ビットコードへの移行を完了することをお勧めします。

AL2023 で 32 ビットバイナリを実行する必要がある場合は、AL2023 上で動作する AL2 コンテナ内の AL2 の 32 ビットユーザースペースを使用できます。

aws-apitools-* から AWS CLI への切り替え

2013 年 9 月に AWS CLI がリリースされる以前から、AWS は、ユーザーが Amazon EC2 API コールを実行できるように、Java に実装された一連のコマンドラインユーティリティを提供していました。これらのツールは 2015 年に廃止され、AWS CLI が Amazon EC2 API をコマンドラインで操作する場合の推奨方法になりました。一連のコマンドラインユーティリティには、次の aws-apitools-* パッケージが含まれています。

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

`aws-apitools-*` パッケージのアップストリームサポートは 2017 年 3 月に終了しました。アップストリームサポートはありませんが、Amazon Linux では `aws-apitools-ec2` などこれらのコマンドラインユーティリティの一部を引き続き提供し、ユーザーの下位互換性を維持しています。AWS CLI はアクティブに保守されており、すべての AWS API を使用する手段となるため、`aws-apitools-*` パッケージよりも堅牢で包括的なツールです。

`aws-apitools-*` パッケージは 2017 年 3 月に廃止され、それ以降、更新プログラムは提供されていません。これらのパッケージのユーザーは、できるだけ早く AWS CLI に移行する必要があります。これらのパッケージは AL2023 には含まれていません。

AL1 では `aws-apitools-iam` および `aws-apitools-rds` パッケージも提供されていましたが、AL1 で廃止され、AL2 以降の Amazon Linux には含まれていません。

AL2 での `upstart` から `systemd` への切り替え

AL2 は、AL1 の `upstart` に代わって `systemd` init システムを採用した最初の Amazon Linux リリースでした。AL1 から Amazon Linux の新バージョンに移行する際には、`upstart` 固有の設定を変更する必要があります。AL1 では `systemd` を使用できないため、`upstart` から `systemd` への移行は、AL2 や AL2023 など、Amazon Linux のより新しいメジャーバージョンへの移行の一環としてのみ行うことができます。

AL2 で廃止され、AL2023 で削除された機能

このセクションでは、AL2 で使用でき、AL2023 では使用できなくなった機能について説明します。

トピック

- [32 ビット x86 \(i686\) パッケージ](#)
- [aws-apitools-* に置き換え AWS CLI](#)
- [awslogs は廃止となり、Amazon CloudWatch Logs エージェントを推奨](#)

- [bzip2 リビジョン制御システム](#)
- [cgroup v1](#)
- [log4j ホットパッチ \(log4j-cve-2021-44228-hotpatch\)](#)
- [lsb_release および system-lsb-core パッケージ](#)
- [mccrypt](#)
- [OpenJDK 7 \(java-1.7.0-openjdk\)](#)
- [Python 2.7](#)
- [rsyslog-openssl による rsyslog-gnutls の置き換え](#)
- [Network Information Service \(NIS\) / yp](#)
- [Amazon VPC create-dhcp-options 内の複数のドメイン名](#)
- [glibc の Sun RPC](#)
- [audit ログの OpenSSH キーフィンガープリント](#)
- [ld.gold リンカー](#)
- [ping6](#)
- [ftp パッケージ](#)

32 ビット x86 (i686) パッケージ

[AL1 の 2014.09 リリース](#)において、これが 32 ビット AMI を提供する最後のリリースになることが発表されました。したがって、[AL1 の 2015.03 リリース](#)以降、Amazon Linux では 32 ビットモードでのシステム実行をサポートしていません。AL2 は x86-64 ホスト上の 32 ビットバイナリのランタイムサポートを限定的に提供しており、新しい 32 ビットバイナリの構築を可能にする開発パッケージも提供していません。AL2023 には 32 ビットユーザースペースパッケージは含まれなくなりました。64 ビットコードへの移行を完了することをお勧めします。

AL2023 で 32 ビットバイナリを実行する必要がある場合は、AL2023 上で動作する AL2 コンテナ内の AL2 の 32 ビットユーザースペースを使用できます。

aws-apitools-* に置き換え AWS CLI

2013 年 9 AWS CLI 月の のリリース以前は、 で実装された AWS 一連のコマンドラインユーティリティを利用できるようになりました。これによりJava、お客様は Amazon EC2 API コールを実行できます。これらのツールは 2015 年に廃止され、コマンドラインから Amazon EC2 APIsを操作する

ための推奨方法 AWS CLI になりました。これには、次の `aws-apitools-*` パッケージが含まれます。

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

`aws-apitools-*` パッケージのアップストリームサポートは 2017 年 3 月に終了しました。アップストリームサポートがないにもかかわらず、Amazon Linux はユーザーに下位互換性を提供するために、これらのコマンドラインユーティリティ (`aws-apitools-ec2` など) の一部を引き続き提供しました。AWS CLI は、`aws-apitools-*` パッケージよりも堅牢で完全なツールであり、アクティブに保守されており、すべての AWS APIs を使用する手段を提供します。

`aws-apitools-*` パッケージは 2017 年 3 月に廃止され、それ以降、更新プログラムは提供されていません。これらのパッケージのすべてのユーザーは AWS CLI、できるだけ早くに移行する必要があります。これらのパッケージは AL2023 には含まれていません。

awslogs は廃止となり、Amazon CloudWatch Logs エージェントを推奨

[awslogs](#) パッケージは AL2 では廃止され、AL2023 では存在しなくなりました。これは、`amazon-cloudwatch-agent` パッケージで利用可能な[統合 CloudWatch Logs エージェント](#)に置き換えられます。詳細については、「[Amazon CloudWatch Logs ユーザーガイド](#)」を参照してください。

bzr リビジョン制御システム

[GNU Bazaar](#) (bzr) リビジョン制御システムは AL2 では廃止され、AL2023 では存在しなくなりました。

bzr のユーザーは、リポジトリを git に移行することをお勧めします。

cgroup v1

AL2023 は Unified Control Group 階層 (cgroup v2) に移行しますが、AL2 は cgroup v1 を使用します。AL2 は cgroup v2 をサポートしていないため、この移行は AL2023 への移行の一環として完了する必要があります。

log4j ホットパッチ (log4j-cve-2021-44228-hotpatch)

Note

log4j-cve-2021-44228-hotpatch パッケージは AL2 では廃止され、AL2023 では削除されています。

[CVE-2021-44228](#) に対応して、Amazon Linux は AL1 と AL2 用の [Apache Log4j 用ホットパッチ](#) の RPM パッケージバージョンをリリースしました。[Amazon Linux へのホットパッチの追加に関する発表](#)の中で、「ホットパッチのインストールは、CVE-2021-44228 または CVE-2021-45046 を軽減する log4j バージョンへの更新に代わるものではない」ことが示されています。

ホットパッチは、log4j へのパッチ適用までの時間を確保するための緩和策でした。AL2023 の最初の一般提供リリースは [CVE-2021-44228](#) の 15 か月後だったため、AL2023 にはホットパッチ (有効化されているかどうかにかかわらず) が含まれていません。

Amazon Linux で独自の log4j バージョンを実行しているお客様は、[CVE-2021-44228](#) または [CVE-2021-45046](#) の影響を受けないバージョンに更新していることを確認することをお勧めします。

lsb_release および system-lsb-core パッケージ

これまで、一部のソフトウェアは (system-lsb-core パッケージによって AL2 で提供されている) lsb_release コマンドを呼び出して、実行されている Linux 配布に関する情報を取得していました。Linux 標準ベース (LSB) ではこのコマンドが導入され、Linux 配布でも採用されました。Linux 配布は、この情報を /etc/os-release およびその他の関連ファイルに保持するという、より単純な標準を使用するように進化しました。

os-release 標準は systemd から生まれました。詳細については、「[systemd os-release ドキュメント](#)」を参照してください。

AL2023 には lsb_release コマンドおよび system-lsb-core パッケージは含まれません。Amazon Linux やその他の主要な Linux 配布との互換性を維持するために、ソフトウェアは os-release 標準への移行を完了する必要があります。

mcrypt

mcrypt ライブラリおよび関連する PHP 拡張機能は AL2 では廃止され、AL2023 には存在しなくなりました。

アップストリームの PHP は、2016年 12 月に最初にリリースされ、[PHP 7.1 において mcrypt 拡張機能を廃止し](#)、2019 年 10 月の最終リリースされたで終了しました。

アップストリームの mcrypt ライブラリは[2007 年に最後にリリース](#)を行いましたが、[SourceForge が 2017 年に新しいコミットに必要とする](#) cvs リビジョン制御からの移行は行われていません。最新のコミット (および 3 年前のみ) は 2011 年のものであり、プロジェクトに管理者がいるという記述は削除されています。

mcrypt を引き続き使用しているユーザーには、コードを OpenSSL に移植することが推奨されています。mcrypt は AL2023 には追加されません。

OpenJDK 7 (java-1.7.0-openjdk)

Note

AL2023 には、Java ベースのワークロードをサポートするために複数のバージョンの [Amazon Corretto](#) が用意されています。OpenJDK 7 パッケージは AL2 では廃止され、AL2023 では存在しなくなりました。AL2023 で利用可能な最も古い JDK は Corretto 8 によって提供されます。

Amazon Linux での Java に関する詳細については、「[AL2023 での Java](#)」を参照してください。

Python 2.7

Note

AL2023 は Python 2.7 を削除したため、Python を必要とする OS コンポーネントはすべて Python 3 で動作するように記述されています。Amazon Linux によって提供されサポートされているバージョンの Python を引き続き使用するには、Python 2 コードを Python 3 に変換します。

Amazon Linux での Python の詳細については、「[AL2023 での Python](#)」を参照してください。

rsyslog-openssl による rsyslog-gnutls の置き換え

rsyslog-gnutls パッケージは AL2 では廃止され、AL2023 では存在しなくなりました。rsyslog-openssl パッケージは、rsyslog-gnutls パッケージの使用をドロップインに置き換える必要があります。

Network Information Service (NIS) / yp

Network Information Service (NIS) は、当初は Yellow Pages または YP と呼ばれていましたが、AL2 では廃止され、AL2023 では存在しなくなりました。これには、ypbind、ypserv および yp-tools のパッケージが含まれます。NIS と統合されている他のパッケージでは、この機能は AL2023 で削除されています。

Amazon VPC create-dhcp-options 内の複数のドメイン名

Amazon Linux 2 では、domain-name パラメータの複数のドメイン名を [create-dhcp-options](#) に渡すことができ、結果として、/etc/resolv.conf に search foo.example.com bar.example.com などが含まれることがありました。Amazon VPC の DHCP サーバーは、DHCP オプション 15 を使用して指定されたドメイン名のリストを送信します。オプション 15 は単一のドメイン名のみをサポートします ([RFC 2132 セクション 3.17 を参照](#))。AL2023 ではネットワーク構成に systemd-networkd を使用しており、これは RFC に準拠しているため、AL2 に存在していた偶発的な機能は AL2023 には存在しません。

[AWS CLI](#) および [Amazon VPC ドキュメント](#) には、「一部の Linux オペレーティングシステムでは、スペースで区切られた複数のドメイン名を使用できます」と記載されています。ただし、Windows や他の Linux オペレーティングシステムでは、この値は単一のドメイン名として処理されるため、予期せぬ動作の原因となります。DHCP オプションセットが、単一のドメイン名として値を処理するオペレーティングシステムを実行するインスタンスを持つ VPC に関連付けられている場合は、ドメイン名を 1 つだけ指定します。

AL2023 などのこれらのシステムでは、DHCP オプション 15 を使用して 2 つのドメインを指定し (1 つのみ許可)、[ドメイン名でスペース文字が無効](#)であるため、スペース文字は 032 としてエンコードされ、結果として /etc/resolv.conf には search foo.example.com032bar.example.com が含まれます。

複数のドメイン名をサポートするには、DHCP サーバーで DHCP オプション 119 を使用する必要があります ([RFC 3397、セクション 2 を参照](#))。Amazon VPC DHCP サーバーでサポートされている場合は、「[Amazon VPC ユーザーガイド](#)」を参照してください。

glibc の Sun RPC

glibc での Sun RPC の実装は AL2 では廃止され、AL2023 では削除されました。Sun RPC 機能が必要な場合は、libtirpc ライブラリ (AL2 および AL2023 で利用可能) の使用に移行することをお勧めします。また、libtirpc を採用することで、アプリケーションは IPv6 をサポートできます。

この変更は、アップストリームの glibc による機能の削除をコミュニティ全体が採用していることを反映しています。[Fedora での glibc からの Sun RPC インターフェイスの削除](#)や [Gentoo での同様の変更](#)がその例です。

audit ログの OpenSSH キーフィンガープリント

AL2 のライフサイクルの後半で、認証に使用されるキーフィンガープリントを出力するためのパッチが OpenSSH パッケージに追加されました。この機能は AL2023 には存在しません。

ld.gold リンカー

ld.gold リンカーは AL2 で使用でき、AL2023 で削除されています。明示的に gold リンカーを参照するソフトウェアを構築するユーザーは、通常の (ld.bfd) リンカーに移行する必要があります。

アップストリームの「[GNU Binutils バージョン 2.44 リリースノート](#)」(2025 年 2 月リリース) には、ld.gold の削除が文書化されています: 「以前の慣行を変更し、このリリースでは binutils-2.44.tar の tarball に gold リンカーのソースは含まれていません。これは、gold リンカーが廃止され、ボランティアが自発的に開発とメンテナンスを継続することを提案しない限り、最終的に削除されるためです。」

ping6

AL2023 では、通常の ping ユーティリティが IPv6 をネイティブにサポートしており、別の /bin/ping6 を必要としません。AL2023 では、/usr/sbin/ping6 は /usr/bin/ping 実行ファイルへのシンボリックリンクです。

この変更は、[Fedora での Ping IPv6 の変更](#)など、新しい iputils のバージョンが提供する機能をコミュニティ全体で導入していることに応じたものです。

ftp パッケージ

AL2 の ftp パッケージは、AL2023 以降の Amazon Linux では使用できなくなりました。これは、セキュリティ、保守性、最新のソフトウェア開発プラクティスに対する継続的なコミットメントの

一環として決定されたものです。AL2023 への移行にあたって (またはそれ以前に)、使用されている `ftp` のレガシーパッケージを代替機能の 1 つに移行することをお勧めします。

背景

`ftp` のレガシーパッケージは、アップストリームで何年もアクティブにメンテナンスされていません。ソースコードの最後の重要な更新は 2000 年代初頭に行われ、元のソースリポジトリは使用できなくなっています。一部の Linux ディストリビューションではセキュリティ脆弱性のパッチが適用されていますが、コードベースはほとんど保守されていません。

推奨される代替機能

AL2023 には、FTP 機能の代わりとして、アクティブに保守されている最新の機能がいくつか用意されています。

`lftp` (AL2 および AL2023 で利用可能)

FTP、HTTP、SFTP、およびその他のプロトコルをサポートする高度なファイル転送プログラム。従来の `ftp` クライアントよりも多くの機能を提供し、アクティブに保守されています。

インストールのコマンド: `dnf install lftp`

`curl` (AL2 および AL2023 で利用可能)

URL を用いたデータ転送のための汎用性の高いコマンドラインツール。FTP、FTPS、HTTP、HTTPSをはじめとする多数のプロトコルをサポートしています。

AL2023 では、デフォルトで `curl-minimal` パッケージを介して使用できます。より広範囲なプロトコルをサポートするには、オプションで `curl-full` を使用して `dnf swap curl-minimal curl-full` にアップグレードできます。

`wget` (AL2 および AL2023 で利用可能)

ウェブからファイルをダウンロードするための非インタラクティブなコマンドラインユーティリティ。HTTP、HTTPS、FTP プロトコルをサポートしています。

インストールのコマンド: `dnf install wget` (すべての AL2023 イメージにデフォルトでインストールされているわけではありません)

`sftp` (AL2 および AL2023 で利用可能)

SSH 経由で動作し、暗号化されたファイル転送を可能にする安全なファイル転送プロトコル。

デフォルトでは、OpenSSH パッケージの一部として提供されます。

移行に関する考慮事項

アプリケーションまたはスクリプトが ftp のレガシークライアントに依存している場合は、次の移行アプローチを検討してください。

1. 最新の代替機能を使用するようにスクリプトを更新する: ftp レガシークライアントの代わりに lftp、curl、wget、または sftp を使用するようにスクリプトを変更します。
2. パッケージの依存関係を確認する: 一部のアプリケーションでは、最新のプロトコルへの移行が内部で行われてからかなり時間が経過しているにもかかわらず、ftp パッケージがパッケージメタデータに依存関係として一覧表示されている場合があります。このような場合、ftp パッケージに /usr/bin/ftp が含まれていなくても、アプリケーションは AL2023 で正しく動作する可能性があります。指定された依存関係のみに依存するのではなく、アプリケーションの実際の要件を確認します。
3. アプリケーションの依存関係を更新する: ftp パッケージとの依存関係を宣言していものの、実際には使用していないアプリケーションについては、パッケージメタデータを更新してこの不要な依存関係を削除します。

セキュリティに関する考慮事項

FTP プロトコルは、認証情報を含むデータをプレーンテキストで送信します。セキュリティ重視のアプリケーションでは、推奨される代替ツールでサポートされている SFTP や HTTPS などの暗号化された代替機能を使用することを強くお勧めします。

AL2023 で廃止

このセクションでは、AL2023 に存在し、将来のバージョンの Amazon Linux で削除される可能性がある機能について説明します。各セクションでは、機能の内容と、Amazon Linux から削除される予定時期について説明します。

Note

このセクションは、Linux エコシステムが進化し、Amazon Linux の将来のメジャーバージョンのリリースに近づくに従って、随時更新されます。

トピック

- [32bit x86 \(i686\) ランタイムのサポート](#)

- [aspell](#)
- [Berkeley DB \(libdb\)](#)
- [cron](#)
- [IMDSv1](#)
- [pcre バージョン 1](#)
- [System V init \(sysvinit\)](#)
- [EOL パッケージが非推奨に](#)

32bit x86 (i686) ランタイムのサポート

AL2023 は、32bit x86 (i686) バイナリを実行する機能を保持します。Amazon Linux の次のメジャーバージョンでは、32bit ユーザースペースバイナリの実行がサポートされなくなります。

aspell

AL2023 には aspell パッケージが同梱されますが、非推奨となり、Amazon Linux の次のメジャーリリースで削除されます。代替の hunspell や enchant2 などに移行することをお勧めします。

AL2023 での aspell の非推奨は、[Fedora での aspell の非推奨](#)など、コミュニティ全体の動向に沿ったものです。

Berkeley DB (libdb)

AL2023 には、Berkeley DB (libdb) ライブラリのバージョン 5.3.28 が同梱されています。これは、制限の緩い Sleepycat ライセンスから GNU Affero GPLv3 (AGPL) ライセンスへと移行する前の Berkeley DB の最終バージョンです。

AL2023 には、Berkeley DB (libdb) に依存するパッケージはほとんどなく、ライブラリは Amazon Linux の次のメジャーリリースで削除されます。

Note

AL2023 の dnf パッケージマネージャーは、Berkeley DB (BDB) 形式の rpm データベースの読み取り専用サポートを保持します。このサポートは、Amazon Linux の次のメジャーリリースで削除されます。

libdb の非推奨は、[Fedora での libdb の非推奨](#)など、コミュニティ全体の動向に追随するものです。

cron

cronie パッケージは AL2 AMI にデフォルトでインストールされ、定期的なタスクをスケジューリングする従来の crontab 方法をサポートしています。AL2023 では、cronie はデフォルトで含まれていません。したがって、crontab のサポートはデフォルトでは提供されなくなりました。

AL2023 では、オプションで cronie パッケージをインストールして元の cron ジョブを使用できます。systemd には機能が追加されているため、systemd タイマーに移行することをお勧めします。

Amazon Linux の将来のバージョン (おそらく次のメジャーバージョン) には、クラシック cron ジョブのサポートが含まれなくなり、systemd タイマーへの移行が完了する予定です。cron からの移行をお勧めします。

IMDSv1

デフォルトでは、AL2023 AMI は IMDSv2 専用モードで起動するように設定され、IMDSv1 の使用が無効になります。IMDSv1 を有効にして AL2023 を使用するオプションは存続します。Amazon Linux の将来のバージョンでは、IMDSv2 専用が適用される可能性があります。

IMDS 設定の詳細については、「Amazon EC2 ユーザーガイド」の「[AMI を設定する](#)」を参照してください。

pcre バージョン 1

レガシーの pcre パッケージは非推奨となり、Amazon Linux の次のメジャーリリースで削除されます。pcre2 パッケージは後継パッケージです。AL2023 の初期バージョンには、pcre 向けに構築された限定数のパッケージが同梱されていましたが、これらのパッケージは AL2023 内で pcre2 に移行されます。非推奨の pcre ライブラリは AL2023 で引き続き使用できます。

Note

非推奨バージョンの pcre は、AL2023 の有効期間全体にわたり、セキュリティ更新プログラムが提供されません。pcre サポートライフサイクルと、パッケージがセキュリティ更新プログラムを受信する期間の詳細については、[pcre パッケージのパッケージサポートに関する声明](#)を参照してください。

pcr の非推奨と pcr2 への移行は、[Fedora での pcr の非推奨](#)など、コミュニティ全体の動向に追随するものです。

System V init (sysvinit)

AL2023 は、System V サービス (init) スクリプトとの下位互換性を保持しますが、アップストリーム systemd プロジェクトでは ([v254 リリース](#)の一環として) [System V サービススクリプトのサポートの廃止](#)を発表し、systemd の将来のバージョンで当サポートを削除することを示しました。詳細については、[を参照してください。systemd](#)

AL2023 は System V サービス (init) スクリプトとの下位互換性を保持しますが、System V サービス (init) スクリプトのサポートが次のメジャーリリースで Amazon Linux から削除される場合に備えて、ネイティブ systemd ユニットファイルの使用に移行することをお勧めします。

EOL パッケージが非推奨に

AL2023 で利用可能な各パッケージには、Amazon Linux 固有の情報に関する[サポートステートメント](#)が付属しています。これらのステートメントは、OS のコア部分とその有効期間、[the section called "PHP"](#) や [the section called "Python"](#) などのパッケージに関連しています。AL2023 は複数のバージョンで提供され、各バージョンはアップストリームのオープンソースプロジェクトと同期間サポートされます。

AL2023 では、dnf パッケージマネージャーを使用して、パッケージサポート情報を取得できます。詳細については、「[パッケージサポート情報の取得](#)」を参照してください。

Amazon Linux のメジャーバージョンが終了する前にパッケージがサポートされなくなった場合、このパッケージは非推奨となり、Amazon Linux の次のメジャーバージョンには含まれないと想定されます。

[the section called "PHP"](#) や [the section called "Python"](#) などのパッケージは、Amazon Linux のメジャーバージョンごとに複数のバージョンが同梱され、バージョンごとにサポートライフサイクルが異なります。これらのパッケージは、Amazon Linux の新しい各メジャーバージョンでも存続し、パッケージのメジャーバージョン間の重複はほとんどないか、まったくないと想定されます。依存関係を選択するときは、Amazon Linux パッケージのサポートタイムラインを念頭に置くことをお勧めします。

AL2 と AL2023 の比較

以下のトピックでは、AL2 と AL2023 の主な違いについて説明します。

AL1、AL2、AL2023 で廃止された機能の詳細については、「[AL2023 の廃止された機能](#)」を参照してください。

トピック

- [追加、更新、削除されたパッケージ](#)
- [各リリースのサポート](#)
- [命名およびバージョニングの変更](#)
- [最適化](#)
- [複数のアップストリームから供給されています。](#)
- [ネットワークシステムサービス](#)
- [パッケージマネージャー](#)
- [cloud-init の使用](#)
- [グラフィカルデスクトップサポート](#)
- [コンパイラトリプレット](#)
- [32 ビット x86 \(i686\) パッケージ](#)
- [lsb_release および system-lsb-core パッケージ](#)
- [Extra Packages for Enterprise Linux \(EPEL\)](#)
- [Python 2.7 は Python 3 に置き換えられました](#)
- [セキュリティ更新](#)
- [安定性向上のための確定的な更新](#)
- [デフォルトの Amazon EBS ボリュームとしての gp3](#)
- [統合コントロールグループ階層 \(cgroup v2\)](#)
- [systemd タイマーによる cron の置き換え](#)
- [ツールチェーンの改善: gcc、binutils、および glibc](#)
- [systemd ジャーナルによる rsyslog の置き換え](#)
- [パッケージの依存関係の最小化](#)
- [デフォルト JVM としての Amazon Corretto](#)
- [AWS CLIV2](#)

- [UEFI 優先ブートと Secure Boot](#)
- [SSH サーバーのデフォルト設定の変更](#)
- [AL2023 のカーネルの変更 \(AL2 との比較\)](#)
- [/tmp は tmpfs になりました](#)
- [AMI とコンテナイメージの変更](#)
- [Amazon Linux 2 AMI と Amazon Linux 2023 AMI にインストールされているパッケージの比較](#)
- [Amazon Linux 2 と Amazon Linux 2023 最小 AMIにインストールされているパッケージの比較](#)
- [Amazon Linux 2 と Amazon Linux 2023 ベースコンテナイメージにインストールされているパッケージの比較](#)

追加、更新、削除されたパッケージ

AL2023 には、何千もの使用可能なソフトウェアパッケージが含まれています。AL2023 で追加、更新、または削除されたすべてのパッケージを以前の Amazon Linux バージョンと比較したリストについては、「[AL2023 でのパッケージ変更](#)」を参照してください。

AL2023 でパッケージを追加または変更するようリクエストするには、GitHub の「[amazon-linux-2023 リポジトリ](#)」に問題を提出してください。

各リリースのサポート

AL2023 については、5 年間のサポートを提供しています。

詳細については、「[リリース頻度](#)」を参照してください。

命名およびバージョンングの変更

AL2023 は、AL2 がプラットフォーム識別用にサポートしているのと同じメカニズムをサポートしています。AL2023 では、プラットフォーム識別用の新しいファイルも導入されています。

詳細については、「[命名およびバージョンング](#)」を参照してください。

最適化

AL2023 は起動時間を最適化して、インスタンス起動からお客様のワークロードの実行までの時間を短縮します。これらの最適化は、Amazon EC2 インスタンスのカーネル設定、cloud-init 設定、および kmod または systemd などの OS のパッケージに組み込まれた機能に及びます。

最適化の詳細については、「[パフォーマンスと運用の最適化](#)」を参照してください。

複数のアップストリームから供給されています。

AL2023 は RPM ベースで、Fedora の複数のバージョンや CentOS 9 Stream などの他の配布から供給されたコンポーネントが含まれています。Amazon Linux カーネルは、他の配布とは別に選んだ kernel.org から直接提供される長期サポート (LTS) リリースから供給されています。

詳細については、「[Fedora との関係](#)」を参照してください。

ネットワークシステムサービス

systemd-networkd システムサービスは AL2023 のネットワークインターフェースを管理します。これは、ISC dhclient または dhcpcd を使用する AL2 からの変更です。

詳細については、「[ネットワークサービス](#)」を参照してください。

パッケージマネージャー

AL2023 のデフォルトのソフトウェアパッケージ管理ツールは DNF です。DNF は YUM の後継で、AL2 のパッケージ管理ツールです。

詳細については、「[パッケージ管理ツール](#)」を参照してください。

cloud-init の使用

AL2023 では、cloud-init はパッケージリポジトリを管理します。Amazon Linux の以前のバージョンでは、cloud-init はデフォルトでセキュリティ更新がインストールされていました。これは AL2023 ではデフォルトではありません。AL2023 が起動時に releasever を更新できるようにする新しい確定的アップグレード機能には、起動時にパッケージ更新を有効にする方法が説明されています。詳細については、「[AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)」および「[安定性向上のための確定的な更新](#)」を参照してください。

AL2023 では、cloud-init および SELinux を使用できます。詳細については、「[cloud-init を使用して enforcing モードを有効にする](#)」を参照してください。

Cloud-init は cloud-init で HTTP(S) を使用してリモートロケーションから設定コンテンツをロードします。以前のバージョンでは、Amazon Linux はリモートリソースが使用できなくなっても警告しません。AL2023 では、リモートリソースが使用できないと致命的なエラーが発生し、cloud-init の実行

に失敗します。AL2 からのこの動作変更により、「フェールクローズ」のデフォルト動作がより安全になりました。

詳細については、「[カスタマイズされた cloud-init](#)」、および「[cloud-init ドキュメント](#)」を参照してください。

グラフィカルデスクトップサポート

AL2023 には、AL2 で使用される MATE デスクトップに代わる、リリース 2023.7 時点の GNOME ベースのグラフィカルデスクトップ環境があります。このバージョンでは、AL2023 のクラウド最適化パフォーマンスを維持しながら、ユーザーに異なるデスクトップエクスペリエンスを提供します。GNOME デスクトップ環境は、さまざまなカスタマイズオプション、システム統合機能、および個別のユーザーインターフェイス設計を提供し、ユーザーに以前の MATE デスクトップ環境に代わるものを提供します。詳細については、[AL2023 グラフィカルデスクトップ](#) ページを参照してください。

コンパイラトリプレット

AL2023 は、GCC および LLVM に amazon がベンダーであることを示すコンパイラトリプレットを設定します。

したがって、AL2 aarch64-redhat-linux-gcc は AL2023 上の aarch64-amazon-linux-gcc になります。

これはほとんどのユーザーには完全に公開されており、AL2023 でコンパイラを構築しているユーザーにのみ影響する可能性があります。

32 ビット x86 (i686) パッケージ

[AL1 の 2014.09 リリース](#)の一環として、32 ビット AMI を提供する最後のリリースになることが発表されました。そのため、[AL1 の 2015.03 リリース](#)以降、Amazon Linux は 32 ビットモードでのシステム実行をサポートしなくなりました。AL2 は x86-64 ホスト上の 32 ビットバイナリのランタイムサポートを限定的に提供しており、新しい 32 ビットバイナリの構築を可能にする開発パッケージも提供していませんでした。AL2023 には 32 ビットユーザースペースパッケージは含まれなくなりました。64 ビットコードへの移行を完了することをお勧めします。

AL2023 で 32 ビットバイナリを実行する必要がある場合は、AL2023 上で動作する AL2 コンテナ内の AL2 の 32 ビットユーザースペースを使用できます。

lsb_release および system-lsb-core パッケージ

これまで、一部のソフトウェアは (system-lsb-core パッケージによって AL2 で提供されている) lsb_release コマンドを呼び出して、実行されている Linux 配布に関する情報を取得していました。Linux 標準ベース (LSB) ではこのコマンドが導入され、Linux 配布でも採用されました。Linux 配布は、この情報を /etc/os-release およびその他の関連ファイルに保持するという、より単純な標準を使用するように進化しました。

os-release 標準は systemd から生まれました。詳細については、「[systemd os-release ドキュメント](#)」を参照してください。

AL2023 には lsb_release コマンドおよび system-lsb-core パッケージは含まれません。Amazon Linux やその他の主要な Linux 配布との互換性を維持するために、ソフトウェアは os-release 標準への移行を完了する必要があります。

Extra Packages for Enterprise Linux (EPEL)

Warning

AL2 epel Extra によりサードパーティーの EPEL7 リポジトリが有効化されました、2024-06-30 の時点で、サードパーティーの EPEL7 リポジトリは維持されなくなりました。

このサードパーティーのリポジトリは 今後更新されません。つまり、EPEL リポジトリ内のパッケージのセキュリティ修正はありません。

このセクションでは、にあるパッケージの AL2023 のオプションについて説明します EPEL。

Extra Packages for Enterprise Linux (EPEL) は、エンタープライズレベルの Linux オペレーティングシステム用の多数のパッケージを作成することを目的とした Fedora コミュニティ内のプロジェクトです。このプロジェクトは主に、RHEL および CentOS パッケージを作成してきました。AL2 は CentOS 7 との互換性が高いのが特徴です。そのため、多くの EPEL7 パッケージが AL2 で動作します。

AL2023 とバイナリ互換性がある EPEL バージョンはありません。ただし、AL2023 で EPEL7 パッケージを使用するお客様は、いくつかのオプションがあります。一部の EPEL パッケージには AL2023 の代替手段があり、他のパッケージは の一部として提供されています [Amazon Linux の補足パッケージ](#)。

⚠ Warning

AL2023 で使用するように設計されたリポジトリのみを追加します。
他のディストリビューション向けに設計されたリポジトリは、現時点では動作する場合がありますが、AL2023 のパッケージ更新や、AL2023 向けに設計されていないリポジトリの更新によって、今後も動作し続ける保証はありません。

このページでは、AL2 で顧客が使用する EPEL7 パッケージとその AL2023 に対応するパッケージについて説明します。

残りのパッケージでは、Amazon Linux (SPAL) のサブリメンタルパッケージを使用できる場合があります。SPAL は Amazon Linux 2023 専用に構築された数千の EPEL9 パッケージを提供していますが、これらのパッケージは AWS Support Plans の対象外です。つまり、SPAL CVEs は追跡されず、パッチはアップストリームで利用可能な場合にのみ提供されます。

⚠ Important

を使用する [Amazon Linux の補足パッケージ](#) 前に、 のドキュメントを参照してください。

トピック

- [axel - HTTP/FTP クライアント](#)
- [brotli および libbrotli - 圧縮](#)
- [collectd - 統計収集デーモン](#)
- [cpulimit - CPU 使用率リミッター](#)
- [exim - メール転送エージェント](#)
- [fuse3 - ユーザースペースのファイルシステム \(FUSE\) v3](#)
- [ganglia - 分散モニタリングシステム](#)
- [git-lfs - Git を使用した大きなファイルのバージョン管理](#)
- [haveged - HAVEGE アルゴリズムを使用したエントロピーソース](#)
- [inotify-tools - inotify コマンドラインツール](#)
- [iperf - TCP/UDP パフォーマンスベンチマーク](#)
- [jemalloc - malloc に代わる実装](#)

- [libbsd - BSD 互換関数ライブラリ](#)
- [libserf - HTTP クライアントライブラリ](#)
- [libzstd - zstd 圧縮ライブラリ](#)
- [lighttpd ウェブサーバー](#)
- [lshell - 制限付きシェル](#)
- [monit - プロセス、ファイル、ディレクトリ、デバイスモニター](#)
- [nodejs](#)
- [perl-Config-General](#)
- [python2-lockfile - ファイルロック](#)
- [python2-rsa - 純粋な Python RSA](#)
- [python2-simplejson - Python 2 の JSON ルーチン](#)
- [rkhunter - ルートキットハンター](#)
- [rssh - OpenSSH で使用するための制限付きシェル](#)
- [sscg - 自己署名 SSL 証明書ジェネレーター](#)
- [stress - ストレステスト](#)
- [stress-ng - ストレステスト](#)
- [tmpwatch - 最終アクセス時間に基づいてファイルを削除する](#)
- [xmlstarlet - コマンドライン XML ユーティリティ](#)

axel - HTTP/FTP クライアント

axel パッケージは EPEL7 に含まれていたものの、Amazon Linux の一部として提供されたことはありません。AL2023 で使用できる代替手段は curl および wget です。

Warning

axel の -S オプションは、ファイルのミラーリングを見つける際に暗号化されていない http を利用します。

axel の利用は curl または wget に移行することを強くお勧めします。

brotli および libbrotli - 圧縮

brotli パッケージと libbrotli パッケージは EPEL7 にあり、brotli パッケージのみが AL2 コアで利用できました。

brotli パッケージと libbrotli パッケージの両方が AL2023 に含まれています。

brotli パッケージは、次のコマンドを使用して AL2023 にインストールできます。

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

libbrotli パッケージは、次のコマンドを使用して AL2023 にインストールできます。

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

collectd - 統計収集デーモン

collect パッケージは EPEL7 に含まれており、collectd および collectd-python3 AL2 Extras でも利用できました。

collectd パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

cpulimit - CPU 使用率リミッター

Amazon Linux 2023 では、systemd がプロセスの CPU 使用率またはプロセスのグループを制限する機能を提供します。この機能は、どの systemd サービスでも簡単に使用できます。

systemd が提供する強力なリソース制御機能を使用して、タスクまたはタスクのグループが消費できるリソースを制限することができます。詳細については、アップストリームの「[systemd.resource-control](#)」ドキュメントと [systemd を使用して AL2023 でのプロセスリソースの使用を制限する](#) を参照してください。

exim - メール転送エージェント

exim パッケージは EPEL7 に含まれており、以前は AL1 で使用できました。Amazon Linux 2023 は、postfix と sendmail メール転送エージェント (MTA) の両方を提供します。

fuse3 - ユーザースペースのファイルシステム (FUSE) v3

fuse3 パッケージ (fuse3-libs と fuse3-devel を含む) は EPEL7 にありました。これらのパッケージは AL2023 の一部であり、それぞれ関連する次のコマンドを実行してインストールできます。

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

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

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

ganglia - 分散モニタリングシステム

ganglia パッケージは EPEL7 に含まれており、以前は AL1 で使用できました。AL2 には同梱されていません。

アップストリームプロジェクトには、活動が停止していた期間があり、その間にいくつかの公開されていた CVE が対処されていませんでした。アップストリームプロジェクトには最近の活動がありましたが、AL2023 に ganglia を追加する予定はありません。

git-lfs - Git を使用した大きなファイルのバージョン管理

git-lfs パッケージは EPEL7 に含まれていました。Amazon Linux 2023 では、git-lfs パッケージはコアリポジトリに含まれています。AL2023 では、git-lfs は以下のコマンドを実行することでインストールできます。

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

haveged - HAVEGE アルゴリズムを使用したエントロピーソース

haveged パッケージは EPEL7 に含まれていました。Amazon Linux 2023 にはエントロピーソースが事前設定されており、haveged を利用する必要はありません。

inotify-tools - inotify コマンドラインツール

inotify-tools パッケージは EPEL7 に含まれていました。AL2023 に含まれています。

Note

AL2023 では、systemd はパスベースのアクティベーションをサポートしています。このアクティベーションは、パスが存在するときや変更されたときなどのイベントに対してアクションを実行するために利用できます。

inotify-tools が使用されている多くの用途は、現在では systemd の systemd パスのアクティベーションを使用して、より信頼性の高い方法でより効果的に実現できるようになりました。詳細については、「[systemd.path](#)」を参照してください。

inotify-tools パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

iperf - TCP/UDP パフォーマンスベンチマーク

iperf のバージョン 2 パッケージは EPEL7 に含まれており、testing AL2 Extra でも利用できました。また、AL1 でも利用できました。

Note

iperf3 パッケージも利用可能であり、iperf のバージョン 3 を提供しています。

iperf パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

jemalloc - malloc に代わる実装

jemalloc パッケージは EPEL7 に含まれており、lamp-mariadb10.2-php7.2 および mariadb10.5 AL2 Extras で利用できました。

jemalloc パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

libbsd - BSD 互換関数ライブラリ

libbsd パッケージは EPEL7 に含まれており、testing AL2 Extra でも利用できました。

libbsd パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

libbsd の開発ファイルは、次のコマンドを実行してインストールします。

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

libserf - HTTP クライアントライブラリ

libserf パッケージは EPEL7 に含まれていました。libserf パッケージは Amazon Linux 2023 で提供されています。以下のコマンドを実行することでインストールできます。

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

libzstd - zstd 圧縮ライブラリ

libzstd パッケージは AL2 コアと EPEL7 に含まれていました。libzstd パッケージも AL2023 の一部です。

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

lighttpd ウェブサーバー

lighttpd パッケージは EPEL7 に含まれており、以前は AL1 で使用できました。Amazon Linux 2023 は、Apache サーバー httpd および nginx ウェブサーバーの両方を提供します。

lshell - 制限付きシェル

lshell パッケージが Amazon Linux の一部として提供されたことはありません。これは EPEL6 で利用可能でした。[lshell の Fedora パッケージングリポジトリ](#)には、EPEL7 または Fedora 30 で[パッケージ化されなかった理由](#)が記載されています。また、[Debian から削除されました](#)。

アップストリームの `lshe11` プロジェクトは[アクティブに保持されておらず](#)、[パッチが適用されていない既知の重要な CVE \(CVE-2016-6902 および CVE-2016-6903\)](#) を含んでいます。

Debian のバグで代替案として提案されている [rssh](#) は、アップストリームでも維持されていません。作成者は修正不可能なセキュリティ問題をその理由として挙げています。

このため、AL2023 への `lshe11` の追加は予定されていません。

monit - プロセス、ファイル、ディレクトリ、デバイスモニター

Amazon Linux 2023 では、`systemd` はサービスをモニタリング、開始、停止、再起動するための幅広い機能を提供します。これには、再起動のレート制限、再起動試行間の間隔、障害発生時の別のサービスの起動が含まれます。詳細については、「[systemd.service](#)」のドキュメントを参照してください。

AL2023 では、`systemd` はパスベースのアクティベーションもサポートしています。これは、パスが存在する場合や変更された場合などのイベントに対してアクションを実行するために利用できます。詳細については、「[systemd.path](#)」を参照してください。

`systemd` ユニットには、成功または失敗に対して実行する依存関係、条件、およびアクションを指定できる一般的な設定オプションがあります。詳細については、「[systemd.unit](#)」ドキュメントを参照してください。

`systemd` が提供する強力なリソース制御機能を利用して、モニタリングタスクで CPU やメモリの過剰な使用を防ぐことができます。詳細については、「[systemd.resource-control](#)」を参照してください。

nodejs

`nodejs` バージョン 16 パッケージは EPEL7 含まれていました。`nodejs` は現時点で AL2023 に含まれています。執筆時点では、`nodejs` のバージョン 18 と 20 の両方が AL2023 で利用可能でした。次のコマンドを使用して `nodejs` 18 を AL2023 にインストールできます。

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

次のコマンドを使用して `nodejs` 20 を AL2023 にインストールできます。

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

perl-Config-General

perl-Config-General パッケージは EPEL7 に含まれており、現時点でAL2023 に含まれていません。perl-Config-General パッケージを AL2023 にインストールするには、以下のコマンドを使用できます。

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

Perl モジュールは、特定の Perl モジュールを提供するパッケージのインストールを DNF にインストールさせることでインストールができます。この方法では、OS パッケージ名ではなく、より使い慣れた Perl モジュール名を使用できます。

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

python2-lockfile - ファイルロック

python2-lockfile パッケージは EPEL7 に含まれており、AL2 には python-lockfile パッケージが含まれていました。AL2023 [Python 2.7 は Python 3 に置き換えられました](#) では、このパッケージの Python 2 版は AL2023 に追加されません。

このパッケージの Python 3 バージョンは AL2023 に含まれています。次のいずれかのコマンドを使用して、python3-lockfile パッケージを AL2023 にインストールできます。

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

Python モジュールは、特定の Python モジュールを提供するパッケージを DNF にインストールさせることでインストールができます。

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

python2-rsa - 純粋な Python RSA

python2-rsa パッケージは EPEL7 に含まれており、AL2 には python2-rsa パッケージが含まれていました。AL2023 [Python 2.7 は Python 3 に置き換えられました](#) では、このパッケージの Python 2 版は AL2023 に追加されません。

このパッケージの Python 3 バージョンは AL2023 に含まれています。次のいずれかのコマンドを使用して、python3-rsa パッケージを AL2023 にインストールできます。

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

Python モジュールは、特定の Python モジュールを提供する パッケージを DNF にインストールすることでインストールができます。

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

python2-simplejson - Python 2 の JSON ルーチン

python2-simplejson パッケージは EPEL7 に含まれていました。AL2023 [Python 2.7 は Python 3 に置き換えられました](#) では、このパッケージの Python 2 版は AL2023 に追加されません。

このパッケージの Python 3 バージョンは AL2023 に含まれています。python3-simplejson パッケージを AL2023 にインストールするには、以下のコマンドを使用できます。

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

Python モジュールは、特定の Python モジュールを提供する パッケージを DNF にインストールすることでインストールができます。

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

rkhunter - ルートキットハンター

rkhunter パッケージは chkrootkit とともに AL2023 に含まれています。

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

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

rsch - OpenSSH で使用するための制限付きシェル

rsch パッケージは EPEL7 に含まれていました。アップストリームの [rsch](#) パッケージは維持されません。作成者は修正不可能なセキュリティ問題をその理由として挙げています。

作成者が修正不可能なセキュリティ問題を挙げているため、rsch の AL2023 への追加は計画されていません。

sscg - 自己署名 SSL 証明書ジェネレーター

sscg パッケージは AL2 コアと EPEL7 に含まれていました。sscg パッケージも AL2023 の一部です。

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

stress - ストレステスト

stress パッケージは EPEL7 に含まれており、AL1 でも利用できました。

stress パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

stress-ng - ストレステスト

stress-ng パッケージは EPEL7 に含まれており、testing AL2 Extra でも利用できました。

stress-ng パッケージは AL2023 に含まれており、次のコマンドを実行してインストールできます。

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

tmpwatch - 最終アクセス時間に基づいてファイルを削除する

Amazon Linux 2023 では、この機能は [systemd-tmpfiles](#) によって提供されます。

xmlstarlet - コマンドライン XML ユーティリティ

xmlstarlet パッケージは EPEL7 に含まれていました。AL2023 では使用できません。

アップストリームパッケージは 9 年以上更新されていません (前回更新は 2014 年 8 月)。さらに 4 年前 (少なくとも 2010 年 7 月以降) に、新しいメンテナンス担当者へのリクエストは応答しなくなりました。このため、AL2023 に xmlstarlet を追加する予定はありません。

Python 2.7 は Python 3 に置き換えられました

AL2 は、AL2 コアパッケージの長期サポート (LTS) の一環として、2025 年 6 月まで Python 2.7 のサポートパッチとセキュリティパッチを提供しています。このサポートは、2020 年 1 月に予定され

ている Python 2.7 のサポート終了というアップストリームの Python コミュニティ宣言よりも延長されています。

AL2 は Python 2.7 に大きく依存している yum パッケージマネージャーを使用しています。AL2023 では、dnf パッケージマネージャーは Python 3 に移行され、Python 2.7 を必要としなくなりました。AL2023 は Python 3 に完全に移行しました。

Note

AL2023 は Python 2.7 を削除したため、Python を必要とする OS コンポーネントはすべて Python 3 で動作するように記述されています。Amazon Linux によって提供されサポートされているバージョンの Python を引き続き使用するには、Python 2 コードを Python 3 に変換します。

Amazon Linux での Python の詳細については、「[AL2023 での Python](#)」を参照してください。

セキュリティ更新

Amazon Linux 2023 は、AL2 に存在する強化をさらに改善しています。詳細については、「[Amazon Linux 2023 でのセキュリティおよびコンプライアンス](#)」を参照してください。AL2 からのカーネル強化の変更の詳細については、「[セキュリティに重点を置いたカーネル設定の変更](#)」を参照してください。

トピック

- [SELinux](#)
- [OpenSSL 3](#)
- [IMDSv2](#)
- [log4j ホットパッチ \(log4j-cve-2021-44228-hotpatch\) の削除](#)

SELinux

デフォルトでは、AL2023 の Security Enhanced Linux (SELinux) は、enabled および permissive モードに設定されています。permissive モードでは、アクセスの拒否は記録されますが、強制ではありません。

SELinux は AL2 では disabled であった Amazon Linux カーネルのセキュリティ機能です。SELinux は、カーネルの主要なサブシステムに必須のアクセスコントロール (MAC) アーキテクチャを提供するカーネル機能とユーティリティのコレクションです。

詳細については、「[AL2023 の SELinux モードの設定](#)」を参照してください。

SELinux リポジトリ、ツール、ポリシーについての詳細は、「[SELinux ノートブック](#)」、「[SELinux ポリシーの種類](#)」および「[SELinux プロジェクト](#)」を参照してください。

OpenSSL 3

AL2023 には Open Secure Sockets Layer version 3 (OpenSSL 3) 暗号化ツールキットが搭載されています。AL2023 は TLS 1.3 および TLS 1.2 ネットワークプロトコルをサポートしています。

デフォルトでは AL2 には OpenSSL 1.0.2 が含まれています。OpenSSL 1.1.1 に対してアプリケーションを構築できます。

OpenSSL の詳細については、「[OpenSSL の移行ガイド](#)」を参照してください。

セキュリティの詳細については、「[セキュリティ更新および機能](#)」を参照してください。

IMDSv2

デフォルトでは、AL2023 AMI で起動するインスタンスはすべて IMDSv2-only を使用する必要があります。コンテナ化されたワークロードをサポートできるようにデフォルトのホップ制限は 2 に設定されます。imds-support パラメータを v2.0 に設定してこれを行います。詳細については、Amazon EC2 ユーザーガイドの「[AMI の設定](#)」を参照してください。

Note

セッショントークンの有効期間は 1 秒から 6 時間の間です。IMDSv2 クエリの API リクエストを送信するアドレスは以下のとおりです。

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

これらの設定は手動で書きし、インスタンスメタデータオプションの起動プロパティを使用して IMDSv1 を有効にできます。IAM コントロールを使用して異なる IMDS 設定を適用することもできます。インスタンスメタデータサービスのセットアップと使用の詳細については、「Amazon EC2 ユーザーガイド」の「[IMDSv2 の使用](#)」、「[新しいインスタンスのインスタンスメタデータオプション](#)」

ンの設定」および「[既存のインスタンスのインスタンスメタデータオプションの変更](#)」を参照してください。

log4j ホットパッチ (log4j-cve-2021-44228-hotpatch) の削除

Note

AL2023 は log4j-cve-2021-44228-hotpatch パッケージに含まれていません。

[CVE-2021-44228](#) に対応して、Amazon Linux は AL1 と AL2 用の [Apache Log4j 用ホットパッチ](#) の RPM パッケージバージョンをリリースしました。[Amazon Linux へのホットパッチの追加に関する発表](#)の中で、「ホットパッチのインストールは、CVE-2021-44228 または CVE-2021-45046 を軽減する log4j バージョンへの更新に代わるものではない」ことが示されています。

ホットパッチは、log4j へのパッチ適用までの時間を確保するための緩和策でした。AL2023 の最初の一般提供 (GA) リリースは [CVE-2021-44228](#) の 15 か月後だったため、AL2023 にはホットパッチ (有効化されているかどうかにかかわらず) が含まれていません。

Amazon Linux で独自の log4j バージョンを実行しているユーザーは、[CVE-2021-44228](#) または [CVE-2021-45046](#) の影響を受けないバージョンに更新していることを確認する必要があります。

AL2023 では、[AL2023 の更新](#) でセキュリティパッチを最新の状態に保つためのガイダンスを提供しています。セキュリティアドバイザリは [Amazon Linux セキュリティセンター](#) で公開されています。

安定性向上のための確定的な更新

バージョン管理されたリポジトリ機能を介した確定的なアップグレードにより、すべての AL2023 AMI はデフォルトで特定のリポジトリバージョンにロックされます。確定的な更新を使用すると、パッケージのバージョンと更新の一貫性を高めることができます。メジャーかマイナーかにかかわらず、各リリースには特定のリポジトリバージョンが含まれます。

AL2023 の新機能では、確定的な更新がデフォルトで有効になっています。これは、AL2 や他の以前のバージョンで使用されていた手動の段階的なロック方法を改良したものです。

詳細については、「[AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)」を参照してください。

デフォルトの Amazon EBS ボリュームとしての gp3

AL2023 AMI と AL2 はどちらもルートファイルシステム上の XFS ファイルシステムを使用します。AL2023 では、ルートデバイスファイルシステムの mkfs オプションが Amazon EC2 向けにさらに最適化されています。AL2023 は他にも多数のファイルシステムをサポートしており、特定の要件に合わせて他のボリュームでも使用できます。

AL2023 AMI はデフォルトで Amazon EBS gp3 ボリュームを使用しますが、AL2 AMI はデフォルトで Amazon EBS gp2 ボリュームを使用します。インスタンスの起動時にボリュームタイプを変更できます。

Amazon EBS ボリュームタイプの詳細については、「[Amazon EBS 汎用ボリューム](#)」を参照してください。

インスタンスの起動の詳細については、「Amazon EC2 ユーザーガイド」の「[インスタンスの起動](#)」を参照してください。

統合コントロールグループ階層 (cgroup v2)

コントロールグループ (cgroup) は、プロセスを階層的に整理し、システムリソースをプロセス間で分散するための Linux カーネルの機能です。コントロールグループは、コンテナランタイムの実装や、さまざまな用途に systemd によって使用されています。

AL2 は cgroupv1 をサポートし、AL2023 は cgroupv2 をサポートします。これは [AL2023 ベースの Amazon ECS AMI を使用したコンテナ化されたワークロードのホスト](#) の場合などの、コンテナ化されたワークロードの実行時に特に顕著です。

AL2023 にはまだ cgroupv1 を使用してシステムを実行させるコードが含まれていますが、これは推奨もサポートもされていないため、Amazon Linux の今後のメジャーリリースで完全に削除される予定です。

「[systemd cgroup 委任ドキュメント](#)」など、[低レベルの Linux カーネルインターフェース](#)に関するドキュメントは豊富です。

コンテナ以外のユースケースは、使用できるシステムリソースに制限を設けた systemd ユニットを作成する場合です。詳細については、「[systemd.resource-control](#)」を参照してください。

systemd タイマーによる cron の置き換え

cronie パッケージは AL2 AMI にデフォルトでインストールされ、定期的なタスクをスケジューリングする従来の crontab 方法をサポートしています。AL2023 では、cronie はデフォルトで含まれていません。したがって、crontab のサポートはデフォルトでは提供されなくなりました。

オプションで cronie パッケージをインストールして元の cron ジョブを使用できます。systemd には機能が追加されているため、systemd タイマーに移行することをお勧めします。

ツールチェーンの改善: gcc、binutils、および glibc

AL2023 には AL2 と同じコアパッケージが多数含まれています。

AL2023 用に以下の 3 つのコアツールチェーンパッケージが更新されました。

パッケージ名	AL2	AL2023
glibc	2.26	2.34
gcc	7.3	11.3
binutils	2.29	2.39

詳細については、「[コアツールチェーンパッケージ glibc、gcc、binutils](#)」を参照してください。

更新されたデフォルトの言語標準など、C、C++、および Fortran 言語ランタイムの詳細については、「[AL2023 での C、C++、および Fortran](#)」を参照してください。

最適化の詳細については、「[パフォーマンスと運用の最適化](#)」を参照してください。

systemd ジャーナルによる rsyslog の置き換え

AL2023 では、ロギングシステムパッケージが AL2 から変更されました。AL2023 はデフォルトでは rsyslog をインストールしないため、/var/log/messages のような AL2 にあったテキストベースのログファイルはデフォルトでは使用できません。AL2023 のデフォルト設定は systemd-journal です。これについては journalctl を使用して確認できます。rsyslog は AL2023 のオプションパッケージですが、新しい systemd ベースの journalctl インターフェースおよび関連

パッケージをお勧めします。詳細については、「[journalctl](#) マニュアルページ」を参照してください。

一般的に使用されるコマンドと systemd journal 相当の syslog コマンドを次の表に示します。

AL2 syslog コマンド	AL2023 systemd journal 相当
<pre>[ec2-user ~]\$ cat /var/log/messages</pre>	<pre>[ec2-user ~]\$ journalctl</pre>
<pre>[ec2-user ~]\$ tail -f /var/log/messages</pre>	<pre>[ec2-user ~]\$ journalctl -f</pre>
<pre>[ec2-user ~]\$ grep foo /var/log/messages</pre>	<pre>[ec2-user ~]\$ journalctl grep foo</pre>

パッケージの依存関係の最小化

Amazon Linux 2023 は、多くのパッケージの依存関係グラフを最小化し、アプリケーションのフットプリントを小さくします。AL2 からの重要な変更には、curl-minimal パッケージと gnupg-minimal パッケージが含まれます。これにより、一般的に使用される機能を維持しながら、必要なパッケージの数が大幅に削減されます。

トピック

- [curl および libcurl のパッケージ変更](#)
- [GNU プライバシーガード \(GNUPG\)](#)

curl および libcurl のパッケージ変更

AL2023 では、curl および libcurl パッケージの一般的なプロトコルと機能が curl-minimal と libcurl-minimal に分離されています。これにより、ほとんどのユーザーのディスク、メモリ、依存関係のフットプリントが削減され、AL2023 AMI とコンテナのデフォルトパッケージとなっています。

gopher:// のサポートなどのために curl の全機能が必要な場合は、以下のコマンドを実行して curl-full および libcurl-full パッケージをインストールします。

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

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

GNU プライバシーガード (GNUPG)

AL2023 は、gnupg2 パッケージの最低限の機能や完全な機能を `gnupg2-minimal` と `gnupg2-full` パッケージに分離しています。デフォルトで `gnupg2-minimal` パッケージのみがインストールされています。これにより、rpm パッケージのデジタル署名を検証するのに必要な最小限の機能が提供されます。

キーサーバからキーをダウンロードする機能など、gnupg2 のより多くの機能を利用するには、`gnupg2-full` パッケージがインストールされていることを確認してください。`gnupg2-full` の `gnupg2-minimal` をスワップするには、以下のコマンドを実行します。

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

デフォルト JVM としての Amazon Corretto

AL2023 には、デフォルトで (かつ唯一の) Java 開発キット (JDK) として [Amazon Corretto](#) が含まれています。AL2023 に含まれるすべての Java ベースパッケージは、すべて Amazon Corretto 17 で構築されています。

AL2 から移行する場合は、AL2 の同等の OpenJDK バージョンから Amazon Corretto にスムーズに移行できます。

AWS CLI v2

AL2023 には AWS CLI バージョン 2 が付属していますが、AL2 には バージョン 1 が付属していません AWS CLI。

UEFI 優先ブートと Secure Boot

デフォルトでは、UEFI ファームウェアをサポートするインスタンスタイプで AL2023 AMI で起動されたインスタンスはすべて UEFI モードで起動します。これは、ブートモード AMI パラメータを

uefi-preferred に設定することで行われます。詳細については、「Amazon EC2 ユーザーガイド」の「[ブートモード](#)」を参照してください。

UEFI Secure Boot をサポートする Amazon EC2 インスタンスタイプでは、Amazon Linux 2023 で Secure Boot を有効にできます。詳細については、「[AL2023 での UEFI セキュアブート](#)」を参照してください。

SSH サーバーのデフォルト設定の変更

AL2023 AMI では、リリース時に生成する sshd ホストキーの種類が変更されました。また、起動時に生成されないように、一部のレガシーキータイプが削除されました。クライアントは、rsa-sha2-256 および rsa-sha2-512 プロトコルをサポートするか、ed25519 キーを使用する ssh-ed25519 が必要です。デフォルトでは、ssh-rsa 署名は無効になっています。

さらに、デフォルトの sshd_config ファイルの AL2023 構成設定には UseDNS=no が含まれます。この新しい設定により、DNS 障害によってインスタンスとの ssh セッションを確立できなくなる可能性が低下します。その対価として、authorized_keys ファイル内の `from=hostname.domain,hostname.domain` ラインのエントリが解決されなくなります。sshd が DNS 名を解決しようとしなくなるため、カンマで区切られた `hostname.domain` のそれぞれの値を IP address に対応する値に変換する必要があります。

詳細については、「[デフォルトの SSH サーバー設定](#)」を参照してください。

AL2023 のカーネルの変更 (AL2 との比較)

AL2023 では、6.1 カーネルが導入され、さらに Amazon Linux をクラウド向けに最適化するための多くの設定変更が導入されています。ほとんどのユーザーにとって、これらの変更は完全に透過的であるはずです。

IPv4 TTL

IPv4 の TTL は `sysctl` を介して設定します。デフォルト値は `/etc/sysctl.d/00-defaults.conf` にあります。この値は、通常の `sysctl` メソッドでカスタマイズできます。詳細については、「`man sysctl`」ページを参照してください。

AL2 では `net.ipv4.ip_default_ttl` 値を 255 に設定していましたが、AL2023 では 127 に設定します。これにより、Amazon Linux のデフォルトは他の主要な Linux ディストリビューションと同等になります。特に必要でない限り、このデフォルトは変更しないでください。

セキュリティに重点を置いたカーネル設定の変更

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_G_ON_DISK_CORRUPTION	n	y	n	y	y	y	y	y	y	y
CONFIG_FAULT_P_MIN_R	4096	4096	4096	4096	65536	65536	65536	65536	65536	65536
CONFIG_VMEM	n	y	n	y	n	n	n	n	n	n
CONFIG_VPORT	n	y	n	y	n	n	n	n	n	n
CONFIG_RTIFY_RCE	n	y	n	y	y	y	y	y	y	y
CONFIG_RDENED_ERCOPY_LLBACK	該当 なし	該当 なし	y	y	該当 なし	該当 なし	該当 なし	該当 なし	該当 なし	該当 なし
CONFIG_IT_ON_OC_DEF_T_ON	該当 なし	該当 なし	n	n	n	n	n	n	n	n

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG IT_ON_ E_DEFA _ON	該当 なし	該当 なし	n	n	n	n	n	n	n	n
CONFIG MMU_DE LT_DMA RICT	該当 なし	該当 なし	該当 なし	該当 なし	n	n	n	n	n	n
CONFIG ISC_AL OAD	y	y	y	y	n	n	n	n	n	n
CONFIG HED_CO	該当 なし	該当 なし	該当 なし	該当 なし	該当 なし	y	該当 なし	y	該当 なし	y
CONFIG HED_ST _END_C K	n	y	n	y	y	y	y	y	y	y
CONFIG CURITY ESG_RE ICT	n	n	n	n	y	y	y	y	y	y
CONFIG CURITY LINUX_ ABLE	y	y	y	y	n	n	該当 なし	該当 なし	該当 なし	該当 なし

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_UFFLE_E_ALLOC_OR	該当なし	該当なし	y	y	y	y	y	y	y	y
CONFIG_AB_FREEST_HARED	n	y	y	y	y	y	y	y	y	y
CONFIG_AB_FREEST_RANDOM	n	n	y	y	y	y	y	y	y	y

x86-64 固有のセキュリティに重点を置いたカーネル設定の変更

CONFIG オプション	AL2/4.14/ x86_64	AL2/5.10/ x86_64	AL2023/6.1/ x86_64	AL2023/6.12/ x86_64	AL2023/6.18/ x86_64
CONFIG_AMD_IOMMU	y	y	y	y	y
CONFIG_AMD_IOMMU_V2	m	m	y	該当なし	該当なし
CONFIG_RANDOMIZE_MEMORY	該当なし	y	y	y	y

aarch64 (ARM/Graviton) 固有のセキュリティに重点を置いたカーネル設定の変更

CONFIG オプション	AL2/4.14/aarch64	AL2/5.10/aarch64	AL2023/6.1/aarch64	AL2023/6.12/aarch64	AL2023/6.18/aarch64
CONFIG_ARM64_PTR_AUTH	該当なし	y	y	y	y
CONFIG_ARM64_PTR_AUTH_KERNEL	該当なし	該当なし	y	y	y
CONFIG_ARM64_SW_TTBR0_PAN	y	y	y	y	y

/dev/mem、/dev/kmem、および /dev/port

Amazon Linux 2023 では、AL2 での設定済みの制限に基づいて /dev/mem と /dev/port (CONFIG_DEVMEM および CONFIG_DEVPORT) が完全に無効になります。

/dev/kmem コードは 5.13 カーネルで Linux から完全に削除されました。AL2 では無効になっていましたが、AL2023 では適用されなくなります。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

FORTIFY_SOURCE

AL2023 では、サポートされているすべてのアーキテクチャで CONFIG_FORTIFY_SOURCE が有効になります。この機能はセキュリティを強化する機能です。コンパイラがバッファサイズを判別して検証できる場合、この機能は一般的な文字列やメモリ関数のバッファオーバーフローを検出できます。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

ラインディシプリン自動ロード (CONFIG_LDISC_AUTOLOAD)

AL2023 カーネルは、CAP_SYS_MODULE アクセス許可を持つプロセスからのリクエストがない限り、TIOCSETD ioctl を使用するソフトウェアなどによるラインディシプリンを自動的にロードしません。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

権限のないユーザーによる dmesg へのアクセス (CONFIG_SECURITY_DMESG_RESTRICT)

デフォルトでは、AL2023 は権限のないユーザーに dmesg へのアクセスを許可しません。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

SELinux selinuxfs の無効化

AL2023 では、ポリシーのロード前に SELinux を無効にするというランタイムメソッドを有効にした非推奨の CONFIG_SECURITY_SELINUX_DISABLE カーネルオプションが無効になります。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

その他のカーネル設定の変更

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG	100	250	100	250	100	100	100	100	100	100
CONFIG	4096	8192	4096	8192	4096	8192	4096	8192	4096	8192
CONFIG NIC_ON PS	y	n	y	n	y	y	y	y	y	y

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_NIC_ONPS_VAL	1	0	1	0	1	1	1	1	該当なし	該当なし
CONFIG_P	m	m	m	m	m	m	m	m	m	m
CONFIG_IP	m	m	m	m	n	n	n	n	n	n
CONFIG_N_PV	該当なし	y	該当なし	n	該当なし	n	該当なし	n	該当なし	n

CONFIG_HZ

AL2023 は、x86-64 および aarch64 プラットフォームの両方で CONFIG_HZ を 100 に設定します。

CONFIG_NR_CPUS

AL2023 では、Amazon EC2 にある CPU コアの最大数に近い数値に CONFIG_NR_CPUS を設定します。

OOPS での パニック

AL2023 カーネルは oops が発生するとパニック状態になります。この機能は、カーネルコマンドラインで oops=panic を起動するのと同じです。

カーネル oops とは、システムのさらなる信頼性に影響を与える可能性のある内部エラーをカーネルが検出したときです。

PPP とスリップのサポート

AL2023 は SLIP プロトコルをサポートしていませんが、最新の AL2023 カーネルで PPP プロトコルをサポートできます。

Xen PV ゲストのサポート

AL2023 は Xen PV ゲストとしての実行をサポートしていません。

カーネルファイルシステムのサポート

AL2 のカーネルがマウントをサポートするファイルシステムにいくつかの変更が加えられたほか、カーネルが解析するパーティションスキームも変更されました。

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_SFS	n	m	n	m	n	n	n	n	n	n
CONFIG_RXRPC	n	m	n	m	n	n	n	n	n	n
CONFIG_D_DISK_EL	y	y	y	y	n	n	n	n	n	n
CONFIG_AMFS	m	m	m	m	n	n	n	n	n	n
CONFIG_AMFS_BLOCKDEV	該当なし	該当なし	y	n	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_CLONE	該当なし	該当なし	n	n	n	n	n	n	n	n
CONFIG_EA	m	n	m	n	n	n	n	n	n	n

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_INTEGRITY	n	m	n	m	m	m	m	m	m	m
CONFIG_LOG_WATCHES	n	n	m	m	m	m	m	m	m	m
CONFIG_SWIOTLB	m	n	m	n	n	n	n	n	n	n
CONFIG_VERIFY	m	n	m	n	m	m	m	m	m	m
CONFIG_RYPTFS	n	m	n	m	n	n	n	n	n	n
CONFIG_FAT_FS	該当なし	該当なし	m	m	m	m	m	m	m	m
CONFIG_T2_FS	n	m	n	m	n	n	n	n	n	n
CONFIG_T3_FS	n	m	n	m	n	n	n	n	該当なし	該当なし
CONFIG_S2_FS	m	m	m	m	n	n	n	n	n	n
CONFIG_SPLUS_FS	n	m	n	m	n	n	n	n	n	n
CONFIG_S_FS	n	m	n	m	n	n	n	n	n	n

CONFIG オプション	AL2/4.1 aarch64	AL2/4.1 x86_64	AL2/5.1 aarch64	AL2/5.1 x86_64	AL2023 1/ aarch64	AL2023 1/ x86_64	AL2023 12/ aarch64	AL2023 12/ x86_64	AL2023 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG S_FS	n	n	n	n	n	n	n	n	n	n
CONFIG M_PART ON	n	y	n	y	n	n	n	n	n	n
CONFIG C_PART ON	n	y	n	y	n	n	n	n	n	n
CONFIG S_V2	n	m	n	m	n	n	n	n	n	n
CONFIG FS_FS	n	m	n	n	n	n	n	n	n	n
CONFIG MFS_FS	n	m	n	m	n	n	n	n	n	n
CONFIG LARIS_ _PARTI N	n	y	n	y	n	n	n	n	n	n
CONFIG UASHFS TD	n	y	n	y	y	y	y	y	y	y
CONFIG N_PART ON	n	y	n	y	n	n	n	n	n	n

Andrew File System (AFS) のサポート

カーネルは afs ファイルシステムをサポートするように構築されなくなりました。AL2 には afs のユーザースペースサポートが同梱されていませんでした。

cramfs のサポート

カーネルは cramfs ファイルシステムをサポートするように構築されなくなりました。AL2023 での後継は squashfs ファイルシステムです。

BSD ディスクラベルのサポート

カーネルは BSD ディスクラベルをサポートするように構築されなくなりました。BSD ディスクラベル付きのボリュームを読み込む必要がある場合は、さまざまな BSD を起動できます。

Device Mapper の変更

AL2023 カーネルに設定されているデバイス Mapper のターゲットにいくつかの変更が加えられました。

eCryptFs のサポート

ecryptfs ファイルシステムは Amazon Linux では廃止されました。ecryptfs のユーザースペースコンポーネントは、AL1 にはありましたが、AL2 では削除されました。AL2023 ではカーネルの構築時に ecryptfs がサポートされなくなりました。

exFAT

exFAT ファイルシステムのサポートは、AL2 の 5.10 カーネルで追加されました。AL2 リリース時の 4.14 カーネルには存在していませんでした。AL2023 は引き続き exFAT ファイルシステムをサポートしています。

ext2、ext3、および ext4 のファイルシステム

AL2023 には CONFIG_EXT4_USE_FOR_EXT2 オプションが同梱されています。これは、ext4 ファイルシステムコードを使用してレガシーの ext2 ファイルシステムを読み取ることを意味します。

CONFIG_GFS2_FS

カーネルは CONFIG_GFS2_FS で構築されなくなりました。

Apple Extended HFS ファイルシステムのサポート (HFS+)

AL2 の場合、x86-64 カーネルのみが `hfsplus` ファイルシステムをサポートするように構築されていました。AL2 の 5.15 カーネルでは、どのアーキテクチャにおいても `hfsplus` はサポートされていません。AL2023 では、Amazon Linux における `hfsplus` のサポートを完全に廃止します。

HFS ファイルシステムのサポート

AL2 の場合、x86-64 カーネルのみが `hfs` ファイルシステムをサポートするように構築されていました。AL2 の 5.15 カーネルでは、どのアーキテクチャにおいても `hfs` はサポートされていません。AL2023 では、Amazon Linux における `hfs` のサポートを完全に廃止します。

JFS ファイルシステムのサポート

古い AL2 x86-64 カーネルは、`jfs` ファイルシステムをサポートするように構築されていました。AL2 の 5.15 カーネルでは、どのアーキテクチャにおいても `jfs` はサポートされていません。AL1 も AL2 も JFS ユーザースペースを同梱していませんでした。AL2023 では、Amazon Linux における `jfs` のサポートを完全に廃止します。

アップストリームの Linux カーネルでは、[JFS の削除を検討](#)しています。したがって、JFS ファイルシステムにデータがある場合は、別のファイルシステムに移行する必要があります。2024 年、JFS は現在のすべての Amazon Linux カーネルから削除されました。

Windows Logical Disk Manager (動的ディスク) のサポート (CONFIG_LDM_PARTITION)

AL2023 では、MS-DOS 形式のパーティションを持つ Windows 2000、Windows XP、または Windows Vista の動的ディスクをサポートしなくなりました。このコードは Windows Vista で導入された新しい GPT ベースの動的ディスクをサポートしていませんでした。

Macintosh パーティションマップのサポート

AL2023 は従来の Macintosh パーティションマップをサポートしなくなりました。最新の macOS バージョンでは、デフォルトでこの古いタイプではなく、最新の GPT パーティションテーブルが作成されます。

NFSv2 のサポート

AL2023 は NFSv2 をサポートしなくなりましたが、NFSv3、NFSv4、NFSv4.1、および NFSv4.2 は引き続きサポートします。NFSv3 以降に移行することをお勧めします。

NTFS (CONFIG_NTFS_FS)

ntfs3 コードでは、AL2 の 5.10 カーネル以降、Amazon Linux での NTFS ファイルシステムにアクセスするための ntfs が置き換えられました。AL2023 には ntfs コードが含まれなくなり、NTFS ファイルシステムへのアクセスは ntfs3 コードのみに依存するようになりました。

romfs ファイルシステム

squashfs ファイルシステムは Amazon Linux の romfs ファイルシステムの後継であり、AL2023 カーネルは romfs をサポートするように構築されなくなりました。

Solaris x86 ハードディスクパーティションフォーマット

AL2023 は Solaris x86 ハードディスクパーティション形式をサポートしなくなりました。

squashfs zstd 圧縮

AL2023 では、すべてのサポート対象アーキテクチャで zstd の圧縮 squashfs ファイルシステムのサポートを追加します。

Sun パーティションテーブルのサポート

AL2023 には Sun パーティションテーブル形式 (CONFIG_SUN_PARTITION) のサポートが含まれなくなりました。

/tmp は tmpfs になりました

Amazon Linux 2023 では、Amazon Linux 2 と比較した場合、/tmp の動作に変更が加えられています。AL2 のデフォルト設定では、/tmp と /var/tmp の両方がルートファイルシステムに存在していました。Amazon Linux 2023 では、/tmp はデフォルトで tmpfs を使用し、RAM の 50% を上限として最大100 万 inodes まで使用できます。これらの変更により、Amazon Linux は他の Linux ディストリビューションの動作と一致するようになります。

AL2023 のファイルシステムレイアウトの詳細については、「[ファイルシステムのレイアウト](#)」セクションの「[/tmp](#)」と「[/var/tmp](#)」を参照してください。

AMI とコンテナイメージの変更

AMI とコンテナに含まれるパッケージにいくつかの変更がありました。

Amazon Linux 2023 は、[the section called “AL2023 最小コンテナイメージ”](#) を導入し、[the section called “必要最低限の AL2023 コンテナイメージの構築”](#) の構築をサポートします。詳細については、「[コンテナでの AL2023 の使用](#)」を参照してください。

Amazon Linux 2 AMI と Amazon Linux 2023 AMI にインストールされているパッケージの比較

Amazon Linux 2 と AL2023 標準 AMI 間での RPM の比較。

パッケージ	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

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

パッケージ	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

パッケージ	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	

パッケージ	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

パッケージ	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

パッケージ	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

パッケージ	AL2 AMI	AL2023 AMI
glibc2	2.56.1	2.74.7
glibc	2.26	2.34
glibc-all-langpacks	2.26	2.34
glibc-common	2.26	2.34
glibc-gconv-extra		2.34
glibc-locale-source	2.26	2.34
glibc-minimal-lang pack	2.26	
gmp	6.0.0	6.2.1
gnupg2	2.0.22	
gnupg2-minimal		2.3.7
gnutls		3.8.0
go-srpm-macros		3.2.0
gpgme	1.3.2	1.15.1
gpm-libs	1.20.7	1.20.7
grep	2.20	3.8
groff-base	1.22.2	1.22.4
grub2	2.06	
grub2-common	2.06	2.06
grub2-efi-aa64	2.06 (aarch64)	
grub2-efi-aa64-ec2	2.06 (aarch64)	2.06 (aarch64)

パッケージ	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

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

パッケージ	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

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

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

パッケージ	AL2 AMI	AL2023 AMI
libmetalink	0.1.3	0.1.3
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnetfilter_connt rack	1.0.6	
libnfnetworklink	1.0.1	
libnfsidmap	0.25	2.5.4
libnhttp2	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

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

パッケージ	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

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

パッケージ	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

パッケージ	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

パッケージ	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

パッケージ	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

パッケージ	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

パッケージ	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	

パッケージ	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

パッケージ	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

パッケージ	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

パッケージ	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	

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

パッケージ	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

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

パッケージ	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	

パッケージ	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

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

パッケージ	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

Amazon Linux 2 と Amazon Linux 2023 最小 AMI にインストールされているパッケージの比較

Amazon Linux 2 と AL2023 Minimal AMI 間での RPM の比較。

パッケージ	AL2 Minimal	AL2023 Minimal
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

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

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

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

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

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

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

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

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

パッケージ	AL2 Minimal	AL2023 Minimal
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnetfilter_conntrack	1.0.6	
libnfnl	1.0.1	
libnftnl	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-file		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

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

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

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

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

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

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

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

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

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

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

パッケージ	AL2 Minimal	AL2023 Minimal
update-motd	1.1.2	2.2
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.30.2	2.37.4
util-linux-core		2.37.4
vim-data	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
which	2.20	2.21
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	1.1.4	
yum-plugin-priorities	1.1.31	
zlib	1.2.7	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2
zstd		1.5.5

Amazon Linux 2 と Amazon Linux 2023 ベースコンテナイメージにインストールされているパッケージの比較

Amazon Linux 2 および AL2023 ベースのコンテナイメージに存在する RPM の比較。

パッケージ	AL2 コンテナ	AL2023 コンテナ
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	

パッケージ	AL2 コンテナ	AL2023 コンテナ
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

パッケージ	AL2 コンテナ	AL2023 コンテナ
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

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

パッケージ	AL2 コンテナ	AL2023 コンテナ
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	

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

パッケージ	AL2 コンテナ	AL2023 コンテナ
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

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

AL1 と AL2023 の比較

以下のトピックでは、[AL2 との比較](#)で取り上げられていない AL1 と AL2023 の主な違いについて説明します。

Note

AL1 は 2023 年 12 月 31 日をもってサポートが終了しており、2024 年 1 月 1 日以降、セキュリティ更新プログラムやバグ修正は一切提供されていません。AL1 のサポート終了およびメンテナンスサポートの詳細については、ブログ記事「[Amazon Linux AMI のサポート終了に関するアップデート](#)」を参照してください。アプリケーションを AL2023 にアップグレードすることをお勧めします。これには 2028 年までの長期サポートが含まれます。

トピック

- [各リリースのサポート](#)
- [init システムとして systemd が upstart を置き換えます。](#)
- [Python 2.6 および 2.7 は Python 3 に置き換えられました](#)
- [最も古い JDK としての OpenJDK 8](#)
- [AL2023 でのカーネルの変更点 \(Amazon Linux 1 \(AL1\) との比較\)](#)
- [Amazon Linux 1 \(AL1\) と Amazon Linux 2023 AMI にインストールされているパッケージの比較](#)
- [Amazon Linux 1 \(AL1\) と Amazon Linux 2023 最小 AMI にインストールされているパッケージの比較](#)
- [Amazon Linux 1 \(AL1\) と Amazon Linux 2023 ベースコンテナイメージにインストールされているパッケージの比較](#)

各リリースのサポート

AL2023 については、リリース日から 5 年間サポートが提供されます。AL1 は、2020 年 12 月 31 日に標準サポートが終了し、2023 年 12 月 31 日にメンテナンスサポートが終了しました。

詳細については、「[リリース頻度](#)」を参照してください。

init システムとして systemd が upstart を置き換えます。

AL2 では、init システムとして upstart が systemd に置き換えられました。AL2023 でも init システムとして systemd を使用しており、systemd の新たな機能がさらに採用されています。

Python 2.6 および 2.7 は Python 3 に置き換えられました

Python 2.6 は AL1 の 2018.03 リリースでサポート終了としてマークされましたが、パッケージはまだインストール可能な状態でリポジトリに存在していました。AL2 には、サポートされている最も古い Python バージョンとして Python 2.7 が付属しており、AL2023 で Python 3 への移行が完了しています。AL2023 リポジトリに Python 2.x バージョンは含まれていません。

Amazon Linux での Python の詳細については、[AL2023 での Python](#) を参照してください。

最も古い JDK としての OpenJDK 8

AL2023 には、デフォルトで (かつ唯一の) Java 開発キット (JDK) として [Amazon Corretto](#) が含まれています。AL2023 に含まれるすべての Java ベースパッケージは、Amazon Corretto 17 で構築されています。

OpenJDK 1.6.0 (java-1.6.0-openjdk) は AL1 バージョン 2018.03 の最初のリリースでサポート終了となり、OpenJDK 1.7.0 (java-1.7.0-openjdk) は 2020 年半ばにサポート終了となりましたが、どちらも AL1 リポジトリで使用可能でした。AL2023 で利用可能な最も古い OpenJDK バージョンは、Amazon Corretto 8 によって提供されている OpenJDK 8 です。

AL2023 でのカーネルの変更点 (Amazon Linux 1 (AL1) との比較)

カーネルライブパッチ

AL2023 と AL2 のどちらにも、カーネルライブパッチ機能のサポートが追加されています。これにより、Linux カーネルの緊急かつ重要なセキュリティ脆弱性に対して、再起動やダウンタイムなしでパッチを適用できます。詳細については、「[AL2023 のカーネルライブパッチ](#)」を参照してください。

カーネルファイルシステムのサポート

AL1 のカーネルがマウントをサポートするファイルシステムにいくつかの変更が加えられたほか、カーネルが解析するパーティションスキームも変更されました。

CONFIG オプション	AL1/4.14/x86_64	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_AVSFS	m	n	n	n	n	n	n
CONFIG_AVRPC	m	n	n	n	n	n	n
CONFIG_BLOCK_DEVICE	y	n	n	n	n	n	n
CONFIG_CAMFS	m	n	n	n	n	n	n
CONFIG_CAMFS_BLOCKDEV	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_DISK_CLONE	該当なし	n	n	n	n	n	n
CONFIG_DISK_ERA	n	n	n	n	n	n	n
CONFIG_DISK_INTEGRITY	m	m	m	m	m	m	m
CONFIG_DISK_LOG_WRITES	n	m	m	m	m	m	m
CONFIG_DISK_SWITCH	n	n	n	n	n	n	n

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_DEBUG_KERNEL	n	m	m	m	m	m	m
CONFIG_CRYPTO_PHE	m	n	n	n	n	n	n
CONFIG_CRYPTO_PHE	該当なし	m	m	m	m	m	m
CONFIG_CRYPTO_T2	m	n	n	n	n	n	n
CONFIG_CRYPTO_T3	m	n	n	n	n	該当なし	該当なし
CONFIG_GPGME	n	n	n	n	n	n	n
CONFIG_HID	m	n	n	n	n	n	n
CONFIG_HID	m	n	n	n	n	n	n
CONFIG_JIT	n	n	n	n	n	n	n
CONFIG_LVM	y	n	n	n	n	n	n
CONFIG_MMC	y	n	n	n	n	n	n

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_NAMES_V2	m	n	n	n	n	n	n
CONFIG_NAMES_FS	m	n	n	n	n	n	n
CONFIG_ROOT_MFS_FS	m	n	n	n	n	n	n
CONFIG_SECURITY_LARIS_X86_PARTITION	y	n	n	n	n	n	n
CONFIG_SECURITY_UASHFS_ZSTD	y	y	y	y	y	y	y
CONFIG_SECURITY_PARTITION_ON	y	n	n	n	n	n	n

セキュリティ重視のカーネル設定の変更

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_BLOCK_ON_DATA_CORRUPTION	y	y	y	y	y	y	y

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_DEBUG_PAGE_FAULT_MMIO_P_MIN_ADDRESS	4096	65536	65536	65536	65536	65536	65536
CONFIG_DEBUG_VMEM	y	n	n	n	n	n	n
CONFIG_DEBUG_VPORT	y	n	n	n	n	n	n
CONFIG_FORTIFY_SOURCE	y	y	y	y	y	y	y
CONFIG_HARDENED_USERCOPY_FALLBACK	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_INIT_ON_ALLOC_DEFAULT_ON	該当なし	n	n	n	n	n	n
CONFIG_INIT_ON_FREE_DEFAULT_ON	該当なし	n	n	n	n	n	n
CONFIG_INIT_MMU_DEFAULT_DMA_SRICT	該当なし	n	n	n	n	n	n

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_LSC_AUTOLOAD	y	n	n	n	n	n	n
CONFIG_SCHED_CORE	該当なし	該当なし	y	該当なし	y	該当なし	y
CONFIG_SCHED_STACK_END_CHECK	y	y	y	y	y	y	y
CONFIG_SECURITY_DEFAULT_RESTRICT	n	y	y	y	y	y	y
CONFIG_SECURITY_SILENT_LINUX_DEFAULT	y	n	n	該当なし	該当なし	該当なし	該当なし
CONFIG_SWAPFLUFFLE_PAGE_ALLOCATION	該当なし	y	y	y	y	y	y
CONFIG_STAB_FREELIST_HARDENED	y	y	y	y	y	y	y

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_SIBAB_FREELEST_RANDOM	n	y	y	y	y	y	y

その他のカーネル設定の変更

CONFIG オプション	AL1/4.14/ x86_64	AL2023/6. 1/ aarch64	AL2023/6. 1/x86_64	AL2023/6. 12/ aarch64	AL2023/6. 12/ x86_64	AL2023/6. 18/ aarch64	AL2023/6. 18/ x86_64
CONFIG_HZ	250	100	100	100	100	100	100
CONFIG_NUMA_CPUS	8192	4096	8192	4096	8192	4096	8192
CONFIG_PANIC_ON_OOPS	n	y	y	y	y	y	y
CONFIG_PANIC_ON_OOPS_VALUE	0	1	1	1	1	該当なし	該当なし
CONFIG_PANIC_TIMEOUT	m	m	m	m	m	m	m
CONFIG_SIBIP	m	n	n	n	n	n	n
CONFIG_XIPV	y	該当なし	n	該当なし	n	該当なし	n

Amazon Linux 1 (AL1) と Amazon Linux 2023 AMI にインストールされているパッケージの比較

AL1 と AL2023 標準 AMI 間での RPM の比較。

パッケージ	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
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	

パッケージ	AL1 AMI	AL2023 AMI
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		2.11
bc	1.06.95	1.07.1
bind-libs	9.8.2	9.18.28
bind-license		9.18.28
bind-utils	9.8.2	9.18.28
binutils	2.27	2.39
boost-filesystem		1.75.0
boost-system		1.75.0
boost-thread		1.75.0
bzip2	1.0.6	1.0.8
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.62	2023.2.68
c-ares		1.19.1
checkpolicy	2.1.10	3.4
chkconfig	1.3.49.3	1.15

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
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	

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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	
fontpackages-files system	1.41	
fonts-srpm-macros		2.0.5
freetype	2.3.11	
fstrm		0.6.1
fuse-libs	2.9.4	2.9.9
gawk	3.1.7	5.1.0
gdbm	1.8.0	
gdbm-libs		1.19
gdisk	0.8.10	1.0.8
generic-logos	17.0.0	
get_reference_source	1.2	
gettext		0.21

パッケージ	AL1 AMI	AL2023 AMI
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
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	

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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	
jemalloc		5.2.1
jitterentropy		3.4.1
jpackage-utils	1.7.5	
jq		1.7.1
json-c		0.14
kbd	1.15	2.4.0
kbd-misc	1.15	2.4.0
kernel	4.14.336	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
kernel-srpm-macros		1.0
kernel-tools	4.14.336	6.1.112

パッケージ	AL1 AMI	AL2023 AMI
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	
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

パッケージ	AL1 AMI	AL2023 AMI
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
libdb		5.3.28
libdhash		0.5.0
libdnf		0.69.0
libeconf		0.4.0
libedit	2.11	3.1
libev		4.33
libevent	2.0.21	2.1.12
libfdisk		2.37.4
libffi	3.0.13	3.4.4
libfido2		1.10.0
libfontenc	1.0.5	
libgcc		11.4.1

パッケージ	AL1 AMI	AL2023 AMI
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	
libidn	1.18	
libidn2	2.3.0	2.3.2
libini_config		1.3.1
libjpeg-turbo	1.2.90	
libkcap1		1.4.0
libkcap1-hmacalc		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

パッケージ	AL1 AMI	AL2023 AMI
libnetfilter_contrack	1.0.4	
libnfnl	1.0.1	
libnfsidmap	0.25	2.5.4
libnhttp2	1.33.0	1.59.0
libnih	1.0.1	
libnl	1.1.4	
libnl3		3.5.0
libpath_utils		0.2.1
libpcap		1.10.1
libpipeline	1.2.3	1.5.3
libpkgconf		1.8.0
libpng	1.2.49	
libpsl	0.6.2	0.21.1
libpwquality	1.2.3	1.4.4
libref_array		0.1.5
librepo		1.14.5
libreport-fs		2.15.2
libseccomp		2.5.3
libselinux	2.1.10	3.4
libselinux-utils	2.1.10	3.4

パッケージ	AL1 AMI	AL2023 AMI
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	
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	
libstoragegmt		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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
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	

パッケージ	AL1 AMI	AL2023 AMI
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	
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

パッケージ	AL1 AMI	AL2023 AMI
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	
ntpd	4.2.8p15	
ntsysv	1.3.49.3	1.15
numactl	2.0.7	
numactl-libs		2.0.14
ocaml-srpm-macros		6
oniguruma		6.9.7.1
openblas-srpm-macros		2
openldap	2.4.40	2.4.57
openssh	7.4p1	8.7p1
openssh-clients	7.4p1	8.7p1
openssh-server	7.4p1	8.7p1
openssl	1.0.2k	3.0.8
openssl-libs		3.0.8

パッケージ	AL1 AMI	AL2023 AMI
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	
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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
perl-subvars		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	0.7.6	
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

パッケージ	AL1 AMI	AL2023 AMI
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
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	

パッケージ	AL1 AMI	AL2023 AMI
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	
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	

パッケージ	AL1 AMI	AL2023 AMI
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	
python27-pystache	0.5.3	
python27-pyattr	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	
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

パッケージ	AL1 AMI	AL2023 AMI
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
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-jsonschemata		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0

パッケージ	AL1 AMI	AL2023 AMI
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
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
python3-pycparser		2.20
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-pytz		2022.7.1

パッケージ	AL1 AMI	AL2023 AMI
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-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

パッケージ	AL1 AMI	AL2023 AMI
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
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	

パッケージ	AL1 AMI	AL2023 AMI
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	
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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
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

パッケージ	AL1 AMI	AL2023 AMI
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
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	
yum-plugin-upgrade-helper	1.1.31	
yum-utils	1.1.31	
zip	3.0	3.0
zlib	1.2.8	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2

パッケージ	AL1 AMI	AL2023 AMI
zstd		1.5.5

Amazon Linux 1 (AL1) と Amazon Linux 2023 最小 AMI にインストールされているパッケージの比較

AL1 と AL2023 Minimal AMI 間での RPM の比較。

パッケージ	AL1 Minimal	AL2023 Minimal
acpid	2.0.19	
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1
amd-ucode-firmware		20210208
audit	2.6.5	3.0.6
audit-libs	2.6.5	3.0.6
authconfig	6.2.8	
awscli-2		2.15.30
basesystem	10.0	11
bash	4.2.46	5.2.15
binutils	2.27	

パッケージ	AL1 Minimal	AL2023 Minimal
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
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

パッケージ	AL1 Minimal	AL2023 Minimal
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
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

パッケージ	AL1 Minimal	AL2023 Minimal
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	
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

パッケージ	AL1 Minimal	AL2023 Minimal
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	
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

パッケージ	AL1 Minimal	AL2023 Minimal
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
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	

パッケージ	AL1 Minimal	AL2023 Minimal
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
kbd-misc	1.15	2.4.0
kernel	4.14.336	6.1.112
kernel-libbpf		6.1.112
kernel-livepatch-r epo-s3		2023.6.20241031
keyutils-libs	1.5.8	1.6.3
kmod	14	29
kmod-libs	14	29
krb5-libs	1.15.1	1.21.3
less	436	608

パッケージ	AL1 Minimal	AL2023 Minimal
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	
libcom_err	1.43.5	1.46.5
libcomps		0.1.20
libcurl	7.61.1	
libcurl-minimal		8.5.0
libdb		5.3.28
libdnf		0.69.0
libeconf		0.4.0
libedit	2.11	3.1
libfdisk		2.37.4

パッケージ	AL1 Minimal	AL2023 Minimal
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
libkcapi-hmaccalc		1.4.0
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.23.2	2.37.4
libnetfilter_connt rack	1.0.4	
libnfnetlink	1.0.1	
libnghttp2	1.33.0	1.59.0
libnih	1.0.1	
libpipeline		1.5.3

パッケージ	AL1 Minimal	AL2023 Minimal
libpsl	0.6.2	0.21.1
libpwquality	1.2.3	1.4.4
librepo		1.14.5
libreport-filessystem		2.15.2
libseccomp		2.5.3
libselenium	2.1.10	3.4
libselenium-utils	2.1.10	3.4
libsemanage	2.1.6	3.4
libsepol	2.1.7	3.4
libsigsigv		2.13
libsmartcols	2.23.2	2.37.4
libsolv		0.7.22
libss	1.43.5	1.46.5
libssh2	1.4.2	
libstdc++		11.4.1
libstdc++72	7.2.1	
libsysfs	2.1.0	
libtasn1	2.3	4.19.0
libtextstyle		0.21
libudev	173	

パッケージ	AL1 Minimal	AL2023 Minimal
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
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

パッケージ	AL1 Minimal	AL2023 Minimal
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	
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

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

パッケージ	AL1 Minimal	AL2023 Minimal
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	
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	

パッケージ	AL1 Minimal	AL2023 Minimal
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	
python27-pyattr	0.5.0	
python27-PyYAML	3.10	
python27-requests	1.2.3	
python27-setuptools	36.2.7	
python27-six	1.8.0	
python27-urlgrabber	3.10	
python27-urllib3	1.24.3	
python3		3.9.16
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-dateutil		2.8.1

パッケージ	AL1 Minimal	AL2023 Minimal
python3-dbus		1.2.18
python3-distro		1.5.0
python3-dnf		4.14.0
python3-dnf-plugins-core		4.3.0
python3-docutils		0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0
python3-jjsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-libselenium		3.4
python3-libsemanage		3.4
python3-markupsafe		1.1.1
python3-netifaces		0.10.6

パッケージ	AL1 Minimal	AL2023 Minimal
python3-oauthlib		3.0.2
python3-pip-wheel		21.3.1
python3-ply		3.11
python3-policycore utils		3.4
python3-prettytable		0.7.2
python3-prompt-too lkit		3.0.24
python3-pycparser		2.20
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-pytz		2022.7.1
python3-pyyaml		5.4.1
python3-requests		2.25.1
python3-rpm		4.16.1.3
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml- clib		0.1.2
python3-setools		4.4.1
python3-setuptools		59.6.0

パッケージ	AL1 Minimal	AL2023 Minimal
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
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

パッケージ	AL1 Minimal	AL2023 Minimal
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
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	

パッケージ	AL1 Minimal	AL2023 Minimal
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
vim-minimal	9.0.2120	9.0.2153
which	2.19	2.21
xfsprogs		5.18.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	

パッケージ	AL1 Minimal	AL2023 Minimal
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

Amazon Linux 1 (AL1) と Amazon Linux 2023 ベースコンテナイメージにインストールされているパッケージの比較

AL1 および AL2023 ベースのコンテナイメージに含まれる RPM の比較。

パッケージ	AL1 コンテナ	AL2023 コンテナ
alternatives		1.15
amazon-linux-repo-cdn		2023.6.20241031
audit-libs		3.0.6
basesystem	10.0	11
bash	4.2.46	5.2.15
bzip2-libs	1.0.6	1.0.8
ca-certificates	2023.2.62	2023.2.68
chkconfig	1.3.49.3	
coreutils	8.22	

パッケージ	AL1 コンテナ	AL2023 コンテナ
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
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

パッケージ	AL1 コンテナ	AL2023 コンテナ
glibc-common	2.17	2.34
glibc-minimal-lang pack		2.34
gmp	6.0.0	6.2.1
gnupg2	2.0.28	
gnupg2-minimal		2.3.7
gpgme	1.4.3	1.15.1
grep	2.20	3.8
gzip	1.5	
info	5.1	
json-c		0.14
keyutils-libs	1.5.8	1.6.3
krb5-libs	1.15.1	1.21.3
libacl	2.2.49	2.3.1
libarchive		3.7.4
libassuan	2.0.3	2.5.5
libattr	2.4.46	2.5.1
libblkid		2.37.4
libcap	2.16	2.48
libcap-ng		0.8.2
libcom_err	1.43.5	1.46.5

パッケージ	AL1 コンテナ	AL2023 コンテナ
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	
libidn2	2.3.0	2.3.2
libmodulemd		2.13.0
libmount		2.37.4
libnghttp2	1.33.0	1.59.0
libpsl	0.6.2	0.21.1
librepo		1.14.5
libreport-filessystem		2.15.2
libselinux	2.1.10	3.4
libsepol	2.1.7	3.4

パッケージ	AL1 コンテナ	AL2023 コンテナ
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
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

パッケージ	AL1 コンテナ	AL2023 コンテナ
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	
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

パッケージ	AL1 コンテナ	AL2023 コンテナ
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	
python27-pygpme	0.3	
python27-pyliblzma	0.5.3	
python27-pyattr	0.5.0	
python27-urlgrabber	3.10	
python3		3.9.16
python3-dnf		4.14.0
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-libcomps		0.1.20

パッケージ	AL1 コンテナ	AL2023 コンテナ
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
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

パッケージ	AL1 コンテナ	AL2023 コンテナ
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-ovl	1.1.31	
yum-plugin-priorities	1.1.31	
yum-utils	1.1.31	
zlib	1.2.8	1.2.11

AL2023 システム要件

このセクションでは、AL2023 を使用するためのシステム要件について説明します。

トピック

- [AL2023 を実行するための CPU 要件](#)
- [AL2023 を実行するためのメモリ \(RAM\) 要件](#)

AL2023 を実行するための CPU 要件

AL2023 コードを実行するには、使用するプロセッサが特定の最小要件を満たしている必要があります。これらの要件を満たしていない CPU で AL2023 を実行しようとすると、コード実行の非常に早い段階で不正な命令エラーが発生する可能性があります。

最小要件は [Amazon EC2 での AL2023](#)、[コンテナでの AL2023](#)、および [Amazon EC2 以外の AL2023](#) に適用されます。

AL2023 の ARM CPU 要件

すべての AL2023 aarch64 (ARM) バイナリは 64 ビット用に構築されています。32 ビットの ARM バイナリは使用できないため、64 ビットの ARM CPU が必要です。

Note

ARM ベースのインスタンスの場合、AL2023 は Graviton2 以降のプロセッサを使用するインスタンスタイプのみをサポートします。AL2023 は A1 インスタンスをサポートしていません。

AL2023 には、暗号化拡張 (ARMv8.2+crypto) を備えた ARMv8.2 準拠のプロセッサが必要です。aarch64 のすべての AL2023 パッケージは、`-march=armv8.2-a+crypto` コンパイラフラグを使用して構築されています。AL2023 コードを古い ARM プロセッサで実行しようとした場合、適切なエラーメッセージを出力するよう努めていますが、最初のエラーメッセージは不正な命令エラーとなる可能性があります。

 Note

AL2023 aarch64 のベース CPU 要件のため、Raspberry Pi 5 より前の Raspberry Pi システムはどれも最小 CPU 要件を満たしていません。

AL2023 の x86-64 CPU 要件

すべての AL2023 x86-64 バイナリは、`-march=x86-64-v2` をコンパイラに渡すことで x86-64 アーキテクチャの x86-64v2 リビジョンに合わせて構築されています。

アーキテクチャの x86-64v2 リビジョンにより、ベースラインの x86-64 アーキテクチャに加えて以下の CPU 機能が追加されます。

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

これは、2009 年以降にリリースされた x86-64 プロセッサにほぼ対応します。例には Intel Nehalem、AMD Jaguar、Atom Silvermont、および VIA Nano と Eden C のマイクロアーキテクチャが含まれます。

Amazon EC2 では、すべての x86-64 インスタンスタイプは M1、C1、M2 インスタンスファミリーを含む x86-64v2 をサポートします。

32 ビット x86 (i686) AL2023 バイナリは構築されません。AL2023 は 32 ビットユーザースペースバイナリの実行を引き続きサポートしていますが、この機能は非推奨となっており、Amazon Linux の今後のメジャーバージョンで削除される可能性があります。詳細については、「[32 ビット x86 \(i686\) パッケージ](#)」を参照してください。

AL2023 を実行するためのメモリ (RAM) 要件

Amazon EC2 .nano ファミリーのインスタンスタイプ (t2.nano、t3.nano、t3a.nano、および t4g.nano) には、512 MB の RAM が搭載されています。これは AL2023 の最小要件です。

Note

512 MB が最小要件ですが、これらのインスタンスタイプはメモリに制約があり、機能やパフォーマンスが制限される場合があります。

AL2023 イメージは、512 MB 未満の RAM を搭載したシステムではテストされていません。AL2023 ベースのコンテナイメージを 512 MB 未満の RAM で実行できるかどうかは、コンテナ化されたワークロードによって異なります。

いくつかの AL2023 リリース間の `dnf upgrade` など、一部のワークロードでは 512 MB 超の RAM が必要になる場合があります。このため、[AL2023.3](#) リリースでは、RAM が 800 MB 未満のインスタンスに対してデフォルトで `zram` を有効化するようになりました。つまり、コンテナ化されたワークロードの場合、このメモリ量を持つ AL2023 インスタンスで正常に動作するワークロードもあれば、このメモリ使用量に制限されたコンテナで実行すると失敗するワークロードもあります。

RAM が 800 MB 未満のインスタンスタイプでは、AL2023 ([AL2023.3](#) 以降) はデフォルトで `zram` ベーススワップを有効にします。メモリが 800 MB 未満の Amazon EC2 インスタンスタイプの例として、t4g.nano、t3a.nano、t3.nano、t2.nano、t1.micro などがあります。AL2023 はオンデマンドでメモリページの圧縮と解凍を行うため、これらのインスタンスタイプではメモリ不足のシナリオが減少します。これにより、圧縮に必要な CPU 使用量を対価として、より多くのメモリを備えたインスタンスタイプを必要とするワークロードが可能になります。

Amazon Linux カーネル

Amazon Linux 2023 (AL2023) は、アップストリーム Linux コミュニティの年間 LTS カーネルに基づいて、毎年第 Q1 に新しい長期サポート (LTS) カーネルをリリースします。新しい LTS カーネルごとに、アップストリーム Linux カーネルから最新のセキュリティ修正、パフォーマンスの向上、ハードウェアサポート、機能が提供されます。

カーネルライフサイクル

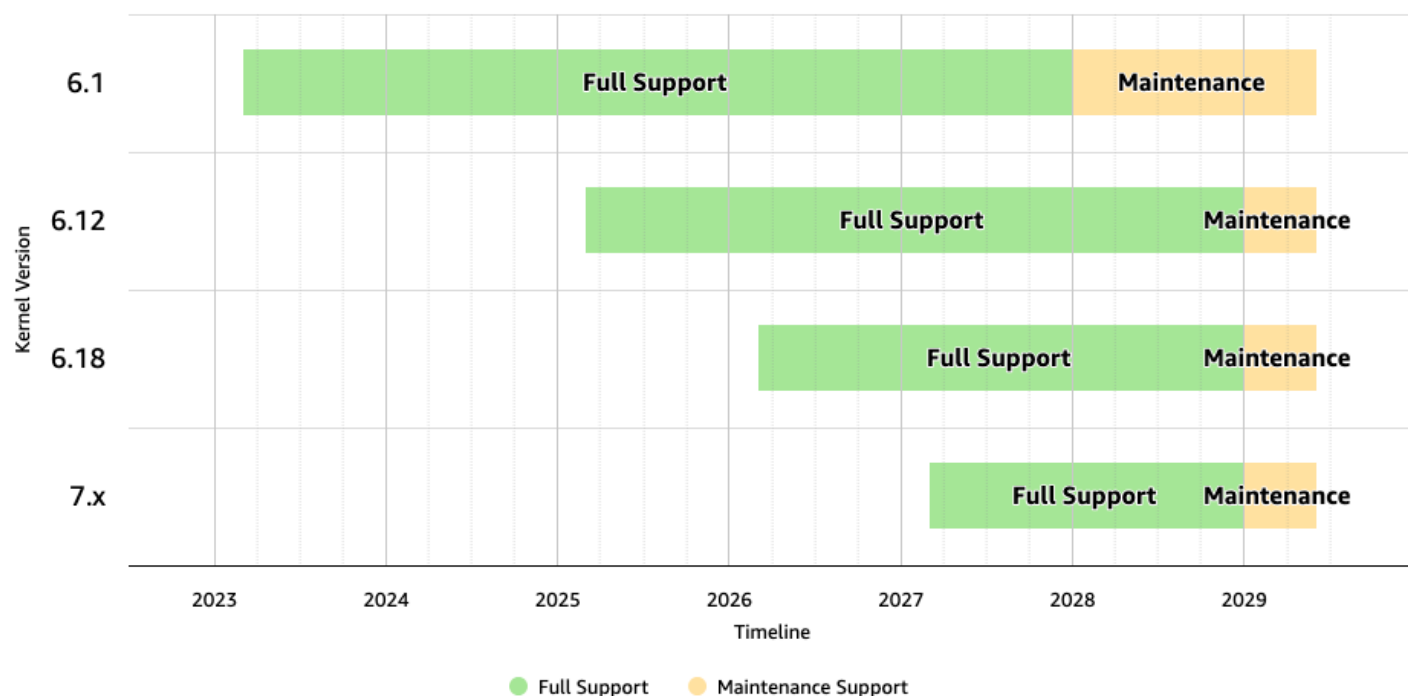
各 AL2023 LTS カーネルは、次の 2 つのフェーズで 4 年間サポートされています。

1. フルサポート (1~2 年) – カーネルは、アップストリーム LTS カーネルの定期的なリベースを通じて、すべての CVE 重要度の修正を受け取ります。このフェーズは、アップストリーム Linux コミュニティの LTS サポートウィンドウと一致しています。アップストリームが LTS サポートウィンドウを拡張する場合、Amazon Linux はそれに従います。
2. メンテナンスサポート (3~4 年) – アップストリーム LTS サポート期間が終了した後も、Amazon Linux チームは、重要で重要な CVEs (CVSS スコア 7.0 以上) および既知の悪用された脆弱性に対する修正を主にバックポートし続けます。このフェーズでは、重要度が低い CVEs はバックポートされません。

4 年後、カーネルはサポートが終了し、セキュリティ更新プログラムを受け取らなくなります。古いカーネルはリポジトリ経由で引き続き使用でき、引き続き使用できます。これ以上のパッチや修正は想定しないでください。この日付より前に、サポートされているカーネルにアップグレードすることをお勧めします。カーネル固有の AMIs は、カーネルのサポート終了日後に更新されなくなります。

次の図は、現在の Amazon Linux LTS カーネルのサポートタイムラインを示しています。

Amazon Linux 2023 Kernel Release Timeline



カーネルの更新

2026 年 6 月以降、AL2023 はデフォルトのカーネルを毎年更新します。AMIs の [al2023-ami-kernel-default](#) セットは最新の LTS カーネルに更新され、新しく起動されたインスタンスに新しいカーネルバージョンが追加されます。

実行中のインスタンスは、新しいカーネルに自動的に更新されません。既存のインスタンスでカーネルをアップグレードするには、新しいカーネルパッケージを明示的にインストールして再起動する必要があります。詳細については、「[AL2023 での Linux カーネルの更新](#)」を参照してください。

最新のセキュリティとパフォーマンスの向上の恩恵を受けるには、新しいカーネルをすぐに採用することを強くお勧めします。古いカーネルでは、時間の経過とともにセキュリティ修正が少なく、遅くなります。カーネルの更新を既存のテストパイプラインとデプロイパイプラインに組み込み、本番環境にロールアウトする前に互換性を検証します。

ディストリビューション間のカーネルバージョンの重複

Amazon Linux ディストリビューション間の移行を簡素化するために、現在および次の Amazon Linux ディストリビューションの両方で少なくとも 1 つのカーネルバージョンを使用できます。これ

により、まず既存のディストリビューションで新しいカーネルにアップグレードし、ワークロードを検証してから、同じカーネルバージョンが利用可能であることを確信して新しいディストリビューションに移行できます。

CVE の処理

AL2023 はカーネル CVEs を次のように処理します。

- 重要かつ重要な CVEs (CVSS 7.0 以降) と既知の悪用された脆弱性は、サポートされているすべてのカーネルにバックポートされます。
- 低および中程度の CVEs (CVSS 7.0 未満) は、フルサポートフェーズ中に通常のアップストリーム LTS リベースを通じて対処されます。メンテナンスサポートフェーズでは、これらの CVEs バックポートされません。低または中程度の CVE が 60 日以内にアップストリーム LTS リベースによって解決されない場合、[Amazon Linux セキュリティセンター](#)では「予定されていない修正」とマークされます。

利用可能な最新のカーネルを実行することは、最新のセキュリティ、パフォーマンス、機能の更新を取得する最善の方法です。

実行すべきこと

1. アプリケーションに継続的インテグレーション (CI) を実装する – 本番稼働用セットアップに変更を加える前に、テスト段階で適切に検証されていることを確認してください。検証にカーネルの更新を含めます。
2. 最新のカーネルを維持する – 新しい年次 LTS カーネルが使用可能になったらすぐに導入します。新しいカーネルは、セキュリティ修正をより迅速に受け取ります。Amazon Linux チームによって推奨されているカーネルバージョンに自動的に適用されるように、[al2023-ami-kernel-default](#)一連の AMIs を使用します。
3. 更新の自動化 – [AWS Systems Manager](#)などのツールを使用して、フリート全体のカーネル更新を管理します。
4. 再起動の計画 – カーネルの更新には再起動が必要です。メンテナンススケジュールに再起動ウィンドウを構築します。

NVIDIA ドライバー

Amazon Linux 2023 は、専用リポジトリを通じて NVIDIA GPU ドライバーと CUDA ツールキット パッケージを提供します。このリポジトリは、[AWS によって管理され、Amazon Linux セキュリティ センター \(ALAS\)](#) を通じてセキュリティアドバイザリを提供します。

トピック

- [NVIDIA リポジトリについて](#)
- [NVIDIA リポジトリの有効化](#)
- [NVIDIA ドライバーのインストール](#)
- [CUDA ツールキットのインストール](#)
- [NVIDIA リポジトリの削除](#)

NVIDIA リポジトリについて

AL2023 NVIDIA リポジトリは、[AL2023 の公式 NVIDIA CUDA リポジトリ](#)からのパッケージをミラーリングします。は、再配布前に AL2023 リリース候補で NVIDIA ソフトウェアを AWS 認定し、このリポジトリ内のパッケージのセキュリティアドバイザリを提供します。

リポジトリは、GovCloud (米国) リージョンや AWS 中国リージョンなど AWS、すべての AWS 商用リージョンで利用できます。

リポジトリは、x86_64 アーキテクチャ用の NVIDIA Tesla (データセンターコンピューティング) とグラフィックスドライバーを提供します。仮想ディスプレイおよびリモートワークステーション機能に使用される GRID ドライバーは含まれません。GRID ドライバーのインストールについては、EC2 [ユーザーガイドの「NVIDIA ドライバーのインストール」](#)を参照してください。

NVIDIA リポジトリの有効化

AL2023 インスタンスで NVIDIA リポジトリを有効にするには、`nvidia-release` パッケージをインストールします。これにより、リポジトリ設定と GPG キーがシステムに追加されます。

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

リポジトリが追加されていることを確認します。


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

amazonlinux-nvidia リポジトリがリストに表示されます。

repo id	repo name
status	
amazonlinux	Amazon Linux 2023 repository
enabled	
amazonlinux-nvidia	Amazon Linux 2023 NVIDIA repository
enabled	

NVIDIA ドライバーのインストール

リポジトリを有効にしたら、 を使用して NVIDIA ドライバーパッケージをインストールできます。dnf。

1. 実行中のカーネルのカーネルヘッダーと開発パッケージをインストールします。

```
[ec2-user ~]$ sudo dnf install kernel-devel-$(uname -r) kernel-headers-$(uname -r) -y
```

2. NVIDIA ドライバーをインストールします。

```
[ec2-user ~]$ sudo dnf install nvidia-driver-cuda -y
```

3. インスタンスを再起動します。

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

4. 再起動後、ドライバーがロードされていることを確認します。

```
[ec2-user ~]$ nvidia-smi
```

CUDA ツールキットのインストール

NVIDIA ドライバーをインストールしたら、CUDA ツールキットをインストールできます。

```
[ec2-user ~]$ sudo dnf install cuda-toolkit -y
```

 Note

NVIDIA Fabric Manager を必要とする GPU インスタンス (P4d、P5、P6 インスタンスタイプなど) の場合は、追加のパッケージをインストールして有効にします。

```
[ec2-user ~]$ DRV_BRANCH="$(modinfo nvidia | grep "^version:" | tr -s ' ' | cut -d ' ' -f 2)"
[ec2-user ~]$ sudo dnf install nvidia-fabricmanager-${DRV_BRANCH} -y
[ec2-user ~]$ sudo systemctl enable --now nvidia-fabricmanager
[ec2-user ~]$ sudo systemctl enable --now nvidia-persistenced
```

Fabric Manager が実行されていて、GPUが NVSwitch を介して接続されていることを確認します。

```
[ec2-user ~]$ sudo systemctl status nvidia-fabricmanager
[ec2-user ~]$ nvidia-smi topo -m
```

トポロジマトリックスでは、GPUにアクティブな NVSwitch 接続を示すNVリンクが表示されます。

インスタンスタイプ固有の要件など、EC2 GPU インスタンスに NVIDIA ドライバーをインストールする詳細な手順については、EC2 [ユーザーガイドの「NVIDIA パブリックドライバ」](#)のインストール」を参照してください。

NVIDIA リポジトリの削除

システムから NVIDIA リポジトリ設定を削除するには:

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

 Important

リポジトリ設定を削除しても、システムに既にインストールされている NVIDIA パッケージは削除されません。

AL2023 グラフィカルデスクトップ

Amazon Linux 2023 では、リリース 2023.7 の時点で、GNOME に基づく軽量でクラウドに最適化されたオプションのグラフィカルインターフェイスを提供しています。この最新のデスクトップ環境は、Firefox などの安全なブラウジングのための組み込みツールで生産性を高める機能を提供するとともに、リモートアクセス用の Amazon DCV と VNC のサポートを維持します。

Note

AL2023 はアップストリームの Firefox 延長サポートリリース (ESR) バージョンに厳密に従い、利用可能になるとすぐに次の ESR に更新されます。詳細については、[Firefox ESR リリースカレンダー](#)と [Firefox リリースノート](#)を参照してください。

関連トピック

グラフィカルデスクトップ環境のインストールの詳細については、以下のドキュメントを参照してください。

- [チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする](#)

Amazon Linux の補足パッケージ

このセクションでは、Amazon Linux (SPAL) のサプリメンタルパッケージを紹介し、パッケージ関連の問題を報告するためのガイドラインとともに、その利点と制限の概要を説明します。

トピック

- [Amazon Linux \(または SPAL\) のサプリメンタルパッケージとは](#)
- [利点](#)
- [SPAL パッケージのサポート](#)
- [パッケージ関連の問題のレポート](#)
- [関連トピック](#)
- [チュートリアル: AL2023 で SPAL リポジトリを設定する](#)

Amazon Linux (または SPAL) のサプリメンタルパッケージとは

Amazon Linux (SPAL) 用サプリメンタルパッケージは、[Enterprise Linux 9 \(EPEL9\) 用の追加パッケージから派生した数千の追加パッケージへのアクセスを提供する専用のパッケージリポジトリです](#)。これらのパッケージは、コア Amazon Linux 2023 で利用可能な既存のソフトウェアを補完します。

SPAL は、AL2023 と互換性のある構築済みのパッケージを提供することでソフトウェアのデプロイを簡素化し、お客様がソースコード自体からパッケージを構築する必要がなくなります。これにより、ソフトウェアのインストールプロセスにかかる時間と労力を節約できます。

Note

SPAL は、リリースバージョンが 2023.9.20251117以降の AL2023 インスタンスで、AWSGovCloud (米国) リージョンや中国を含むすべてのAWS商用リージョンで利用できます。

利点

SPAL は、Amazon Linux 2023 ユーザーにいくつかの主な利点を提供します。

- AL2023 ユースケースの拡張 — pandoc、GDAL、などのコア AL2023 リポジトリ以外の一般的な追加パッケージにアクセスできるため drbd-utils、お客様は複数のビジネスおよび開発ニーズに対応できます。
- AL2023 パッケージ管理の簡素化 — AL2023 用に構築されたパッケージを提供することで、ソースから追加のパッケージを構築するプロセスがなくなり、時間を節約し、コンパイルエラーのリスクが軽減されます。
- AL2 から AL2023 への移行の合理化 — SPAL リポジトリを使用すると、EPEL7 パッケージに依存するワークロードを含め、AL2 から AL2023 へのワークロードのシームレスな移行が可能になります。

SPAL パッケージのサポート

SPAL パッケージは、Amazon Linux 2023 の存続期間中サポートを受けるコア AL2023 パッケージと同じレベルのサポート対象外です。

Important

SPAL を使用する前に、お客様は以下の考慮事項を慎重に評価する必要があります。

- SPAL パッケージはサポートプランの対象外です。AWS
- SPAL パッケージは、アップストリーム EPEL9 から「現状有姿」で提供されます。EPEL9
- SPAL パッケージは CVE セキュリティ追跡を受信しませんAWS。
- SPAL パッケージは、利用可能な場合、アップストリーム EPEL9 からのみセキュリティパッチとバグ修正を受け取ります。

パッケージ関連の問題のレポート

SPAL パッケージで問題が発生した場合は、まずアップストリーム EPEL9 リポジトリ内の対応するパッケージで同じ問題が発生しているかどうかを確認することをお勧めします。これについては、アップストリーム EPEL リポジトリの[問題のリスト](#)を参照してください。

問題が EPEL9 に存在しない場合は、[Amazon Linux 2023 GitHub リポジトリ](#)に問題を作成します。これは、問題が SPAL パッケージのビルドまたは設定に固有のものであることを示しているためです。

このアプローチにより、問題が適切な保守者によって対処され、SPAL パッケージとアップストリーム EPEL パッケージの両方の全体的な品質に貢献します。

Note

報告された問題はベストエフォートベースで処理されます。

関連トピック

システムで SPAL を設定する方法については、次のドキュメントページを参照してください。

- [チュートリアル: AL2023 で SPAL リポジトリを設定する](#)

チュートリアル: AL2023 で SPAL リポジトリを設定する

Amazon Linux 用サプリメンタルパッケージ (SPAL) は、AL2023 用の追加のパッケージリポジトリで、何千ものオープンソースパッケージへのアクセスを顧客に提供します。

次のチュートリアルは、AL2023 インスタンスで SPAL リポジトリを設定するのに役立ちます。リポジトリをインストールすることで、SPAL で利用可能なすべての RPM パッケージにアクセスできます。インストールしたら、パッケージマネージャーを使用して、これらのパッケージをシステムにインストールして使用できます。

内容

- [前提条件](#)
- [前提条件の確認](#)
- [システムに SPAL をインストールする](#)
- [SPAL パッケージのインストール](#)
- [SPAL ソースパッケージのダウンロード](#)
- [SPAL debuginfo パッケージのインストール](#)
- [システムから SPAL リポジトリをアンインストールする](#)
- [関連トピック](#)

前提条件

このチュートリアルでは、AL2023 リリースバージョン 2023.9.20251117以降を使用してインスタンスを既に起動していることを前提としています。詳細については、「[Amazon EC2 での AL2023](#)」および「[AL2023 の更新](#)」のページを参照してください。

前提条件の確認

- インスタンスが前提条件を満たしていることを確認するには、システムにsystem-releaseインストールされている のバージョンを確認できます。

パッケージのバージョンを確認するには、次のコマンドを使用します。

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

コマンドは、メジャーバージョンを含むパッケージに関する情報を表示します。

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

Note

の最新バージョンsystem-releaseがインストールされていることを確認します。sudo dnf upgrade を実行して最新バージョンに更新できます。

システムに SPAL をインストールする

- システムに spal-releaseパッケージをインストールします。これにより、.repo設定ファイルと GPG キーがシステムに追加されます。

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

Note

インストール中に、サポートステートメントが表示されます。このステートメントでは、SPAL のサポート範囲と制限について説明します。この情報をよく確認してください。

2. SPAL リポジトリ設定がシステムに正常に追加されたことを確認します。

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

システムに設定されている 3 つのリポジトリ `amazonlinux-spal`、`amazonlinux-spal-source`、`amazonlinux-spal-debuginfo` が表示されます。

を実行して、設定されたリポジトリのリストを確認することもできます `dnf repolist`。

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

Note

有効と無効の両方のリポジトリを表示するには、`--all` フラグが必要です。

3 つの SPAL リポジトリがすべて使用可能である必要があります。Amazon Linux 2023 SPAL リポジトリ - ソースパッケージと Amazon Linux 2023 SPAL リポジトリ - デバッグリポジトリはデフォルトで無効になっていることに注意してください。

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

SPAL パッケージのインストール

- `dnf install` コマンドを実行して、システムに SPAL パッケージをインストールします。

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

Note

を使用して `dnf list`、SPAL パッケージの完全なリストを表示できます。

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

Note

SPAL はバージョン管理されたリポジトリです。パッケージの最新のリストを表示するには、の最新バージョン `system-release` がインストールされていることを確認してください。

確定的な更新の詳細については、「」を参照してください。 [AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)

SPAL ソースパッケージのダウンロード

RPM ソース (SRPM) パッケージは、主にデベロッパーがソースからパッケージを構築するために使用します。SPAL ソースリポジトリはデフォルトで無効になっています。DNF は、ソースパッケージを必要とするコマンドを使用する場合に自動的に有効にします。

SPAL パッケージのソース RPM をダウンロードするには、次のコマンドを実行します。

```
[ec2-user ~]$ dnf download --source package
```

SPAL debuginfo パッケージのインストール

debuginfo リポジトリには、debuginfo パッケージと debugsource パッケージの両方が含まれています。これらのパッケージは、アプリケーションのデバッグとプロファイリングに役立つデバッグ

記号とソースファイルを提供します。SPAL debuginfo リポジトリはデフォルトで無効になっています。DNF は、デバッグパッケージを必要とするコマンドを使用する場合に自動的に有効にします。

SPAL パッケージの debuginfo パッケージと debugsource パッケージをインストールするには、次のコマンドを実行します。

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

システムから SPAL リポジトリをアンインストールする

1. dnf remove コマンドを使用して SPAL リポジトリ設定を削除します。

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

2. dnf repolist コマンドを実行して、リポジトリが削除されたことを確認します。

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

Important

システムから SPAL リポジトリ設定を削除しても、システムにインストールされている SPAL パッケージは削除されません。

関連トピック

Amazon Linux リポジトリのサブリメンタルパッケージの詳細については、次のドキュメントを参照してください。

- [Amazon Linux の補足パッケージ](#)

AL2023 でのアプリケーションの実行

このセクションでは、Amazon Linux 2023 (AL2023) でアプリケーションを実行する方法について説明し、起動 (および再起動) のタイミングの管理やリソース使用量の制御についても取り上げます。

トピック

- [systemd を使用して AL2023 でのプロセスリソースの使用を制限する](#)
- [cgroups を使用して AL2023 でプロセスのリソース使用量を制限する](#)

systemd を使用して AL2023 でのプロセスリソースの使用を制限する

Amazon Linux 2023 (AL2023) では、systemd を使用して、プロセス (またはプロセスのグループ) で使用できるリソースを制御することをお勧めします。systemd の使用は、cgroups の手動操作や、[cpulimit](#) (以前にサードパーティの [EPEL](#) リポジトリで提供されていた Amazon Linux 専用の機能) などのユーティリティの使用に代わる、強力で使いやすい代替手段となります。

包括的な情報については、アップストリームの「systemd」ドキュメントで「[systemd.resource-control](#)」を参照するか、AL2023 インスタンスに関する man ページで「systemd.resource-control」を参照してください。

以下の例では、stress-ng CPU ストレステストを (stress-ng パッケージから) 使用して CPU を大量に使用するアプリケーションをシミュレーションします。また、memcached を使用してメモリを大量に使用するアプリケーションをシミュレートします。

1 回限りのコマンドに CPU 制限を設定する方法と、サービスにメモリ制限を設定する方法を以下の例で示します。systemd が提供するほとんどのリソース制約は、systemd によってプロセスを実行するあらゆる場所で使用でき、複数の制約を同時に使用できます。以下の例では、わかりやすいように 1 つの制約のみを使用しています。

1 回限りのコマンドを実行するための **systemd-run** によるリソースコントロール

systemd は一般的にシステムサービスに関連付けられていますが、非ルートユーザーによるサービスの実行、タイマーのスケジュール設定、または 1 回限りのプロセスの実行にも使用できます。

以下の例では、サンプルアプリケーションとして `stress-ng` を使用します。最初の例では、`ec2-user` デフォルトアカウントで `systemd-run` を使用してアプリケーションを実行し、2 番目の例ではアプリケーションの CPU 使用率を制限します。

Example リソースの使用量を制限せずに、コマンドラインで `systemd-run` を使用してプロセスを実行する

1. この例で使用する `stress-ng` パッケージが、インストール済みであることを確認します。

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

2. `systemd-run` を使用して、10 秒の CPU ストレステストを実行します。使用できる CPU の量は制限しません。

```
[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` オプションは、ログインしているユーザーとしてコマンドを実行するように `systemd-run` に指示します。`--tty` オプションは、TTY がアタッチされていることを意味し、`--wait` は、サービスが終了するまで待機することを意味します。`--property=CPUAccounting=1` オプションは、プロセスの実行に使用された CPU 時間を記録するように `systemd-run` に指示します。`--property` コマンドラインオプションを使用して、`systemd.unit` 設定ファイルで構成できる `systemd-run` 設定を渡すことができます。

CPU に負荷をかけるように指示すると、`stress-ng` プログラムは使用可能なすべての CPU 時間を使用して、指示した実行時間が経過するまで、テストを実行します。実際のアプリケーションでは、プロセスの合計実行時間を制限することが望ましい場合があります。次の例では、`systemd-run` を使用して設定した上限時間よりも長い時間実行するように `stress-ng` に指示します。

Exampleコマンドラインで **systemd-run** を使用してプロセスを実行し、CPU 使用時間を 1 秒に制限する

1. この例を実行するには、**stress-ng** がインストール済みであることを確認します。
2. **LimitCPU** プロパティは **ulimit -t** と同等であり、このプロセスで使用できる CPU の最大時間を制限します。この例では、10 秒のストレス実行を要求し、CPU 使用時間を 1 秒に制限しているため、コマンドは **SIGXCPU** シグナルを受信して失敗します。

```
[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
```

より一般的には、特定のプロセスで消費できる CPU 時間の割合を制限したい場合があります。次の例では、**stress-ng** が消費できる CPU 時間の割合を制限します。実際のサービスでは、バックグラウンドプロセスが消費できる CPU 時間の最大割合を制限して、ユーザーリクエストを処理するプロセスのためにリソースを解放することが望ましい場合があります。

Example **systemd-run** を使用して 1 つの CPU における CPU 時間の 10% にプロセスを制限する

1. この例を実行するには、**stress-ng** がインストール済みであることを確認します。
2. **CPUQuota** プロパティを使用して、実行するコマンドの CPU 使用率を制限するように **systemd-run** に指示します。プロセスの実行時間を制限するのではなく、使用できる CPU の量のみを制限します。

```
[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
```

CPU アカウンティングを参照すると、サービスの実行時間は 10 秒でも、実際の CPU 時間は 1 秒しか消費されなかったことがわかります。

CPU、メモリ、ネットワーク、IO のリソース使用量を制限するように systemd を設定する方法は数多くあります。包括的なドキュメントについては、アップストリームの「systemd」ドキュメントで「[systemd.resource-control](#)」を参照するか、AL2023 インスタンスに関する man ページで「systemd.resource-control」を参照してください。

バックグラウンドでは、systemd は cgroups などの Linux カーネルの機能を使用して、これらの制限を実装し、手動で設定する必要がないようにしています。[cgroup-v2 に関する Linux カーネルのドキュメント](#)には、cgroups 作業に関する多くの詳細が記載されています。

systemd サービスでのリソースコントロール

システムリソースの使用を制御するために、systemd サービスの [Service] セクションに追加できるパラメータがいくつかあります。これらには、ハード制限とソフト制限の両方が含まれます。各オプションの正確な動作については、アップストリームの「systemd」ドキュメントで「[systemd.resource-control](#)」を参照するか、AL2023 インスタンスに関する man ページで「systemd.resource-control」を参照してください。

一般的に使用される制限は、メモリ使用量のスロットリング制限を指定する MemoryHigh、ハード上限 (上限に達すると OOM キラーが呼び出されます) を設定する MemoryMax、および CPUQuota (前のセクションの説明を参照) です。また、固定値ではなく、重みと優先順位を設定することもできます。

Example **systemd** を使用してサービスにメモリ使用量の制限を設定する

この例では、**memcached** (単純なキー値キャッシュ) のハードメモリ使用量の制限を設定し、OOM Killer をシステム全体ではなく、該当サービスに対してどのように呼び出すかを示します。

1. まず、この例に必要なパッケージをインストールする必要があります。

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

2. **memcached.service** を有効にしてサービスを起動し、**memcached** が実行されるようにします。

```
[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. **memcached.service** が実行されていることを確認します。

```
[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. **memcached** がインストールして実行したら、ランダムなデータをキャッシュに挿入し、それが機能することを確認できます。

/etc/sysconfig/memcached で、**CACHESIZE** 変数はデフォルトで 64 (64 メガバイト) に設定されます。最大キャッシュサイズよりも多くのデータをキャッシュに挿入すると、キャッシュが満杯になり、**memcached-tool** によって一部の項目が削除されることがわかります。また、**memcached.service** が約 64 MB のメモリを使用していることもわかります。

```
[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. MemoryMax プロパティを使用して memcached.service のハード制限を設定します。上限に達すると、OOM Killer が呼び出されます。サービスに追加のオプションを設定するには、オーバーライドファイルにオプションを追加します。これは、/etc/systemd/system/memcached.service.d/override.conf ファイルを直接編集して行うか、systemctl の edit コマンドを使用してインタラクティブに行うことができます。

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

オーバーライドに以下を追加して、サービスのメモリのハード制限を 32 MB に設定します。

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

6. 設定を再ロードするように systemd に指示します。

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


7. memcached.service が 32 MB のメモリ制限で実行されていることを確認します。

```
[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. サービスは、32 MB 未満のメモリを使用している間は正常に機能します。これをチェックするには、32 MB 未満のランダムデータをキャッシュにロードし、サービスのステータスを確認します。

```
[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. これで 32 MB を超えるメモリを memcached で使用できるようになり、memcached のデフォルト設定である 64 MB のフルキャッシュを試すことができます。

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

上記のコマンドの実行中に、memcached サーバーへの接続エラーが発生したことがわかります。これは、OOM Killer で設定した制限により、プロセスが強制終了されたためです。システムの残りの部分は通常どおり機能します。OOM Killer は、制限した memcached.service のみを対象とし、他のプロセスは考慮しません。

```
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
disabled)
   Drop-In: /etc/systemd/system/memcached.service.d
           ##override.conf
   Active: failed (Result: oom-kill) since Fri 2025-01-31 23:20:28 UTC; 2s ago
 Duration: 2.901s
   Process: 360130 ExecStart=/usr/bin/memcached -p ${PORT} -u ${USER} -m
${CACHE_SIZE} -c ${MAXCONN} $OPTIONS (code=killed, signal=KILL)
  Main PID: 360130 (code=killed, signal=KILL)
    CPU: 94ms
```

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

cgroups を使用して AL2023 でプロセスのリソース使用量を制限する

[systemd によるリソースコントロール](#) を使用することが推奨されますが、このセクションでは、基本の `libcgroup-tools` ユーティリティを使ってプロセスの CPU とメモリの使用量を制限する基本的な方法を説明します。どちらも、以前 [EPEL](#) に含まれていた [cpulimit](#) ユーティリティの代わりに使用できる手段です。

以下の例では、`libcgroup-tools` パッケージのユーティリティと `sysfs` の調整可能なオプションを使用して CPU とメモリの使用量を制限しながら、(`stress-ng` パッケージから) `stress-ng` ストレステストを実行する方法について説明します。

コマンドラインで **`libcgroup-tools`** を使用してリソース使用量を制限する

1. `libcgroup-tools` パッケージをインストールします。

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

2. `memory` および `cpu` コントローラーを使用して `cgroup` を作成し、名前 (`our-example-limits`) を付けます。 `-a` および `-t` オプションを使用して、`ec2-user` ユーザーが `cgroup` の調整可能なオプションを制御できるようにします。

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

`/sys/fs/cgroup/our-example-limits/` ディレクトリが追加されており、調整可能な各オプションの制御に使用できるファイルが含まれています。

Note

Amazon Linux 2 では、AL2023 で使用される `cgroup-v1` ではなく `cgroup-v2` を使用します。AL2 では `sysfs` のパスが異なり、`ec2-user` が所有する `/sys/fs/cgroup/memory/our-example-limits` および `/sys/fs/cgroup/cpu/our-example-limits` ディレクトリに、`cgroup` の制限を制御するために使用できるファイルが格納されています。

3. `cgroup` のすべてのプロセスのメモリ使用量を 1 億バイトに制限します。

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

Note

Amazon Linux 2 では、Amazon Linux 2023 が使用する cgroup-v2 ではなく cgroup-v1 を使用します。つまり、調整可能なオプションが一部異なります。AL2 でメモリ使用量を制限するには、代わりに以下の調整可能なオプションが使用されます。

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

4. cgroup のすべてのプロセスの CPU 使用率を 10% に制限します。cpu.max ファイルの形式は \$MAX \$PERIOD で、グループの使用量が \$PERIOD につき \$MAX に制限されます。

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

Amazon Linux 2 では、Amazon Linux 2023 が使用する cgroup-v2 ではなく cgroup-v1 を使用します。つまり、CPU 使用率の制限方法など、調整可能なオプションが一部異なります。

5. 以下の例では、our-example-limits cgroup で stress-ng (dnf install -y stress-ng を実行してインストール可能) を実行します。stress-ng コマンドの実行中に、top コマンドを使用して、使用率が CPU 時間の 10% に制限されていることを確認できます。

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

6. cgroup を削除してクリーンアップします。

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

[Linux カーネルの cgroup-v2 のドキュメント](#)には、その仕組みに関する詳細な情報が含まれています。[CPU](#) および [メモリ](#) コントローラーのドキュメントでは、調整可能な各オプションの使用方法について詳しく説明しています。

での AL2023 の使用 AWS

AL2023 は、他の AWS のサービスと使用するように設定できます。例えば、[Amazon Elastic Compute Cloud](#) (Amazon EC2) インスタンスを起動する際に、AL2023 AMI を選択できます。

これらのセットアップ手順では、AWS Identity and Access Management (IAM) サービスを使用します。IAM の詳細については、以下の資料を参照してください。

- [AWS Identity and Access Management \(IAM\)](#)
- [IAM ユーザーガイド](#)

トピック

- [の開始方法 AWS](#)
- [Amazon EC2 での AL2023](#)
- [コンテナでの AL2023 の使用](#)
- [AWS Elastic Beanstalk での AL2023](#)
- [AWS CloudShell での AL2023 の使用](#)
- [AL2023 ベースの Amazon ECS AMI を使用したコンテナ化されたワークロードのホスト](#)
- [AL2023 での Amazon Elastic File System の使用](#)
- [AL2023 上に構築された Amazon EMR の使用](#)
- [での AL2023 の使用 AWS Lambda](#)

の開始方法 AWS

にサインアップする AWS アカウント

の使用を開始するには AWS、が必要です AWS アカウント。の作成の詳細については AWS アカウント、「AWS アカウント管理 リファレンスガイド」の「[の開始方法 AWS アカウント](#)」を参照してください。

プログラムによるアクセス権を付与する

ユーザーが の AWS 外部で を操作する場合は、プログラムによるアクセスが必要です AWS マネジメントコンソール。プログラムによるアクセスを許可する方法は、 がアクセスするユーザーのタイプによって異なります AWS。

ユーザーにプログラムによるアクセス権を付与するには、以下のいずれかのオプションを選択します。

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	(推奨) コンソール認証情報を一時的な認証情報として使用して AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、AWS Command Line Interface「ユーザーガイド」のAWS「ローカル開発用ログイン」を参照してください。 • AWS SDKs「SDK およびツールリファレンスガイド」のAWS「ローカル開発のためのログイン」を参照してください。AWS SDKs
ワークフォースアイデンティティ (IAM アイデンティティセンターで管理されているユーザー)	一時的な認証情報を使用して AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、「AWS Command Line Interface ユーザーガイド」の「使用する AWS CLI ように AWS IAM アイデンティティセンターを設定する」を参照してください。 • AWS SDKs、ツール、API については、AWS APIs「SDK およびツールリファレンスガイド」の「IAM アイデンティティセンター認証」を参照してください。AWS SDKs

プログラムによるアクセス権を必要とするユーザー	目的	方法
IAM	一時的な認証情報を使用して AWS CLI、AWS SDKs、または AWS APIs。	「IAM ユーザーガイド 」の「 AWS リソースでの一時的な認証情報の使用 」の手順に従います。
IAM	(非推奨) 長期認証情報を使用して AWS CLI、AWS SDKs、または AWS APIs。	使用するインターフェイスの指示に従ってください。 • については AWS CLI、 「AWS Command Line Interface ユーザーガイド」の「IAM ユーザー認証情報を使用した認証」を参照してください。 • AWS SDKs 「SDK とツールリファレンスガイド」の「長期認証情報を使用した認証」を参照してください。 AWS SDKs • API AWS APIs 「IAM ユーザーガイド」の「IAM ユーザーのアクセスキーの管理」を参照してください。

Amazon EC2 での AL2023

AL2023 AMI を使用して Amazon EC2 インスタンスを起動するには、以下のいずれかの手順に従います。標準 AMI または最小 AMI を選択できます。標準 AMI と最小 AMI の違いの詳細については、「[AL2023 標準 \(デフォルト\) と最小 AMI の比較](#)」を参照してください。

トピック

- [Amazon EC2 コンソールを使用した AL2023 の起動](#)
- [SSM パラメータとを使用して AL2023 を起動する AWS CLI](#)

- [を使用して最新の AL2023 AMI を起動する CloudFormation](#)
- [特定の AMI ID を使用して AL2023 を起動する](#)
- [AL2023 AMI の非推奨とライフサイクル](#)
- [AL2023 インスタンスへの接続](#)
- [AL2023 標準と最小の AMI の比較](#)

Amazon EC2 コンソールを使用した AL2023 の起動

Amazon EC2 コンソールを使用して、AL2023 AMI を起動します。

Note

ARM ベースのインスタンスの場合、AL2023 は Graviton2 以降のプロセッサを使用するインスタンスタイプのみをサポートします。AL2023 は A1 インスタンスをサポートしていません。

Amazon EC2 コンソールから AL2023 AMI を使用した Amazon EC2 インスタンスを起動するには、以下の手順に従います。

AL2023 AMI で EC2 インスタンスを起動する方法

1. Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
2. ナビゲーションペインで [AMI] を選択します。
3. ドロップダウンメニューから [パブリックイメージ] を選択します。
4. [検索] フィールドに **al2023-ami** を入力します。

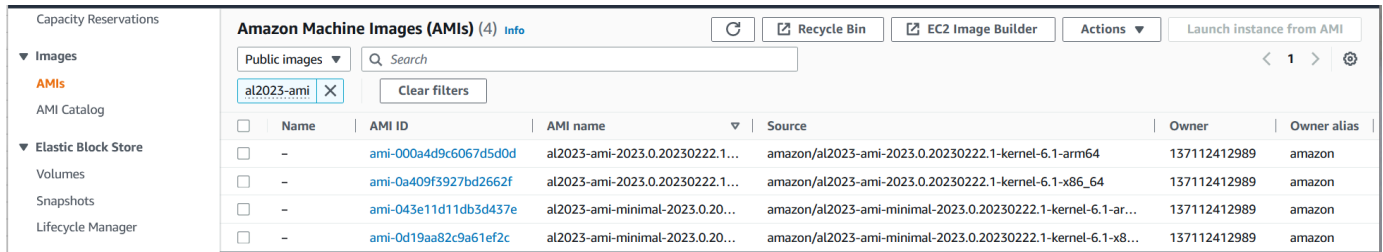
Note

[オーナーエイリアス] 列に Amazon が表示されていることを確認します。

5. リストから AMI を選択します。[ソース] で、AMI を標準または最小にするかを決定できます。AL2023 AMI 名は、以下の形式を使用して解釈できます。

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


6. 以下のイメージは、AL2023 AMI のリストの一部を示しています。



The screenshot shows the Amazon Machine Images (AMIs) console. The left sidebar has a navigation menu with 'Images' selected. The main area displays a table of AMIs. The table has columns for Name, AMI ID, AMI name, Source, Owner, and Owner alias. The first four rows are visible, all starting with 'al2023-ami-'.

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

Amazon EC2 インスタンスの起動方法の詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EC2 Linux インスタンスの使用を開始する](#)」を参照してください。

SSM パラメータと を使用して AL2023 を起動する AWS CLI

では AWS CLI、AMI の SSM パラメータ値を使用して AL2023 の新しいインスタンスを起動できます。具体的には、以下のリストにある動的 SSM パラメータ値のいずれかを使用し、SSM パラメータ value/ の前に /aws/service/ami-amazon-linux-latest/ を追加します。これを使用して、AWS CLI内でインスタンスを起動できます。

- arm64 アーキテクチャの al2023-ami-kernel-default-arm64
- arm64 アーキテクチャ用 (最小 AMI) の al2023-ami-minimal-kernel-default-arm64
- x86_64 アーキテクチャの al2023-ami-kernel-default-x86_64
- x86_64 アーキテクチャ (最小 AMI) の al2023-ami-minimal-kernel-default-x86_64

Note

###各項目はサンプルパラメータです。それらを、ユーザー自身の情報に置き換えます。

```
$ 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 フラグは SSM パラメータ値を指定します。

--instance-type フラグはインスタンスのタイプとサイズを指定します。このフラグは、選択した AMI タイプと互換性がある必要があります。

--region フラグは、インスタンスを作成する AWS リージョン を指定します。

--key-name フラグは、インスタンスへの接続に使用される AWS リージョンのキーを指定します。インスタンスを作成したリージョンに存在するキーを指定しない場合、SSH を使用してインスタンスに接続することはできません。

--security-group-ids フラグは、インバウンドとアウトバウンドのネットワークトラフィックのアクセス権限を決定するセキュリティグループを指定します。

Important

では、ポート 経由でリモートマシンからインスタンスへのアクセスを許可する既存のセキュリティグループを指定 AWS CLI する必要がありますTCP:22。セキュリティグループを指定しない場合、新しいインスタンスはデフォルトのセキュリティグループに配置されます。デフォルトのセキュリティグループでは、インスタンスは VPC 内の他のインスタンスとのみ接続できます。

詳細については、「AWS Command Line Interface ユーザーガイド」の「[Amazon EC2 インスタンスの起動、リスト化、終了](#)」を参照してください。

を使用して最新の AL2023 AMI を起動する CloudFormation

を使用して AL2023 AMI を起動するには CloudFormation、次のいずれかのテンプレートを使用します。

Note

x86_64 と Arm64 AMI にはそれぞれ異なるインスタンスタイプが必要です。詳細については、「[Amazon EC2 のインスタンスタイプ](#)」を参照してください。

JSON テンプレート:

```
{  
  "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 テンプレート:

```

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

```

必要に応じて、「デフォルト」セクションの末尾にある AMI パラメータを置き換えてください。以下のパラメータ値を使用できます。

- arm64 アーキテクチャの al2023-ami-kernel-6.1-arm64
- arm64 アーキテクチャ用 (最小 AMI) の al2023-ami-minimal-kernel-6.1-arm64
- x86_64 アーキテクチャの al2023-ami-kernel-6.1-x86_64
- x86_64 アーキテクチャ (最小 AMI) の al2023-ami-minimal-kernel-6.1-x86_64

以下は動的カーネルの仕様です。デフォルトのカーネルバージョンは、カーネルバージョンがメジャー更新されるたびに自動的に変更されます。

- arm64 アーキテクチャの al2023-ami-kernel-default-arm64
- arm64 アーキテクチャ用 (最小 AMI) の al2023-ami-minimal-kernel-default-arm64
- x86_64 アーキテクチャの al2023-ami-kernel-default-x86_64
- x86_64 アーキテクチャ (最小 AMI) の al2023-ami-minimal-kernel-default-x86_64

特定の AMI ID を使用して AL2023 を起動する

AMI ID を使用して特定の AL2023 AMI を起動できます。Amazon EC2 コンソールの AMI リストを確認することで、どの AL2023 AMI ID が必要かを判断できます。または、[AMI ID を使用して AWS Systems Manager を起動する](#)を使用できます AWS Systems Manager。Systems Manager を使用している場合は、前のセクションに記載されているエイリアスから AMI エイリアスを選択します。詳細については、[「Parameter Store を使用した最新の Amazon Linux AMI IDs AWS Systems Manager」](#)を参照してください。

AL2023 AMI の非推奨とライフサイクル

AL2023 の新しいリリースにはそれぞれ、新しい AMI が含まれています。AMI が登録されると、廃止日が表示されます。各 AL2023 AMI の非推奨日は、[AL2023 のカーネルライブパッチ](#) が個々のカーネルリリースに設定されている期間に合わせて、リリース日から 90 日です。

Note

90 日間の廃止日は個々の AMI を指し、AL2023 [リリース頻度](#) や製品サポート期間を指すものではありません。

AMI 非推奨の詳細については、「Amazon EC2 ユーザーガイド」の「[AMI を非推奨にする](#)」を参照してください。

更新された AMI を定期的に使用してインスタンスを起動することで、更新されたカーネルを含む最新のセキュリティ更新でインスタンスが起動します。AMI の以前のバージョンを起動して更新を適用した場合、インスタンスに最新のセキュリティ更新が適用されない期間があります。最新の AMI を使用するには、SSM パラメータを使用することをお勧めします。

SSM パラメータを使用したインスタンスの起動方法の詳細については、以下を参照してください。

- [SSM パラメータとを使用して AL2023 を起動する AWS CLI](#)
- [を使用して最新の AL2023 AMI を起動する CloudFormation](#)

AL2023 インスタンスへの接続

SSH または AWS Systems Manager を使用して AL2023 インスタンスに接続します。

SSH を使用したインスタンスへの接続

SSH を使ってインスタンスに接続する手順については、「Amazon EC2 ユーザーガイド」の「[SSH クライアントを使用して Linux インスタンスに接続する](#)」を参照してください。

を使用してインスタンスに接続する AWS Systems Manager

AWS Systems Manager を使用して AL2023 インスタンスに接続する方法については、Amazon EC2 ユーザーガイド」の「[セッションマネージャーを使用して Linux インスタンスに接続する](#)」を参照してください。

Amazon EC2 Instance Connect の使用

最小の AMI を除く AL2023 AMI には、EC2 Instance Connect エージェントがデフォルトでインストールされています。最小の AMI から起動した AL2023 インスタンスで EC2 Instance Connect を使用するには、`ec2-instance-connect` パッケージをインストールする必要があります。EC2 Instance Connect の使用手順については、「Amazon EC2 ユーザーガイド」の「[EC2 Instance Connect を使用して Linux インスタンスに接続する](#)」を参照してください。

AL2023 標準と最小の AMI の比較

Amazon EC2 インスタンスは、標準 (デフォルト) または最小の AL2023 AMI で起動できます。標準または最小 AMI タイプの Amazon EC2 インスタンスの起動手順については、「[Amazon EC2 での AL2023](#)」を参照してください。

標準 AL2023 AMI には、最も一般的に使用されるアプリケーションとツールがすべてインストールされています。すぐに使い始めたいが AMI のカスタマイズは不要な場合は、標準 AMI をお勧めします。

最小の AL2023 AMI は、オペレーティングシステム (OS) の実行に必要な最も基本的なツールとユーティリティのみを含む、基本的で簡素化されたバージョンです。OS のフットプリントをできるだけ小さくしたい場合は、最小 AMI を使用することをお勧めします。最小 AMI では、ディスク容量の使

用率がわずかに低下し、長期的なコスト効率が向上します。OS を小さくしたいが、ツールやアプリケーションを手動でインストールしてもかまわない場合は、最小 AMI が適しています。

コンテナイメージは、パッケージセット内の AL2023 最小 AMI に近いものです。

Amazon Linux 2023 イメージにインストールされているパッケージの比較

AL2023 AMI、Minimal AMI、コンテナイメージに存在する RPM の比較。

パッケージ	AMI	Minimal AMI	コンテナ
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)	

パッケージ	AMI	Minimal AMI	コンテナ
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
aws-cfn-bootstrap	2.0		
awscli-2	2.15.30	2.15.30	
basesystem	11	11	11
bash	5.2.15	5.2.15	5.2.15
bash-completion	2.11		
bc	1.07.1		
bind-libs	9.18.28		
bind-license	9.18.28		
bind-utils	9.18.28		
binutils	2.39		
boost-fil esystem	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

パッケージ	AMI	Minimal AMI	コンテナ
ca-certificates	2023 年 2 月 68 日	2023 年 2 月 68 日	2023 年 2 月 68 日
c-ares	1.19.1		
checkpolicy	3.4	3.4	
chkconfig	1.15		
chrony	4.3	4.3	
cloud-init	22.2.2	22.2.2	
cloud-init-cfg-ec2	22.2.2	22.2.2	
cloud-utils-growpart	0.31	0.31	
coreutils	8.32	8.32	
coreutils-common	8.32	8.32	
coreutils-single			8.32
cpio	2.13	2.13	
cracklib	2.9.6	2.9.6	
cracklib-dicts	2.9.6	2.9.6	
crontabs	1.11		
crypto-policies	20220428	20220428	20220428
crypto-policies-scripts	20220428		

パッケージ	AMI	Minimal AMI	コンテナ
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	
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	

パッケージ	AMI	Minimal AMI	コンテナ
dnf-utils	4.3.0		
dosfstools	4.2		
dracut	055	055	
dracut-config-ec2	3.0	3.0	
dracut-config-generic	055	055	
dwz	0.14		
dyninst	10.2.1		
e2fsprogs	1.46.5	1.46.5	
e2fsprogs-libs	1.46.5	1.46.5	
ec2-hibinit-agent	1.0.8		
ec2-instance-connect	1.1		
ec2-instance-connect-selinux	1.1		
ec2-utils	2.2.0	2.2.0	
ed	1.14.2		
efi-filesystem	5	5	
efi-srpm-macros	5		
efivar	38	38	

パッケージ	AMI	Minimal AMI	コンテナ
efivar-libs	38	38	
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	

パッケージ	AMI	Minimal AMI	コンテナ
gettext	0.21	0.21	
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	

パッケージ	AMI	Minimal AMI	コンテナ
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-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		

パッケージ	AMI	Minimal AMI	コンテナ
inih	49	49	
initscripts	10.09	10.09	
iproute	6.10.0	6.10.0	
iputils	20210202	20210202	
irqbalance	1.9.0	1.9.0	
jansson	2.14	2.14	
jemalloc	5.2.1		
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		

パッケージ	AMI	Minimal AMI	コンテナ
keyutils-libs	1.6.3	1.6.3	1.6.3
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

パッケージ	AMI	Minimal AMI	コンテナ
libconfig	1.7.2		
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	

パッケージ	AMI	Minimal AMI	コンテナ
libkcapi-hmaccalc	1.4.0	1.4.0	
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

パッケージ	AMI	Minimal AMI	コンテナ
libseccomp	2.5.3	2.5.3	
libselinux	3.4	3.4	3.4
libselinux- utils	3.4	3.4	
libsemanage	3.4	3.4	
libsepol	3.4	3.4	3.4
libsigsegv	2.13	2.13	2.13
libsmartcols	2.37.4	2.37.4	2.37.4
libsolv	0.7.22	0.7.22	0.7.22
libss	1.46.5	1.46.5	
libsss_certmap	2.9.4		
libsss_idmap	2.9.4		
libsss_ns s_idmap	2.9.4		
libsss_sudo	2.9.4		
libstdc++	11.4.1	11.4.1	11.4.1
libstoragemgmt	1.9.4		
libtalloc	2.3.4		
libtasn1	4.19.0	4.19.0	4.19.0
libtdb	1.4.7		
libtevent	0.13.0		

パッケージ	AMI	Minimal AMI	コンテナ
libtextstyle	0.21	0.21	
libtirpc	1.3.3		
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		

パッケージ	AMI	Minimal AMI	コンテナ
lz4-libs	1.9.4	1.9.4	1.9.4
man-db	2.9.3	2.9.3	
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		

パッケージ	AMI	Minimal AMI	コンテナ
ntsysv	1.15		
numactl-libs	2.0.14	2.0.14	
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		

パッケージ	AMI	Minimal AMI	コンテナ
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	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		

パッケージ	AMI	Minimal AMI	コンテナ
perl-Getopt-Std	1.12		
perl-HTTP-Tiny	0.078		
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		

パッケージ	AMI	Minimal AMI	コンテナ
perl-Pod-Usage	2.01		
perl-POSIX	1.94		
perl-Scalar-List-Utills	1.56		
perl-SelectSaver	1.02		
perl-Socket	2.032		
perl-srpm-macros	1		
perl-Storable	3.21		
perl-subsub	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		

パッケージ	AMI	Minimal AMI	コンテナ
pkgconf-pkg-config	1.8.0		
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-awscli	0.19.19	0.19.19	
python3-babel	2.9.1	2.9.1	
python3-cffi	1.14.5	1.14.5	
python3-chardet	4.0.0	4.0.0	
python3-colorama	0.4.4	0.4.4	

パッケージ	AMI	Minimal AMI	コンテナ
python3-c onfigobj	5.0.6	5.0.6	
python3-c ryptography	36.0.1	36.0.1	
python3-daemon	2.3.0		
python3-d ateutil	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-d ocutils	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-j mespath	0.10.0	0.10.0	
python3-j sonpatch	1.21	1.21	

パッケージ	AMI	Minimal AMI	コンテナ
python3-j sonpointer	2.0	2.0	
python3-j sonschema	3.2.0	3.2.0	
python3-l ibcomps	0.1.20	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16	3.9.16
python3-l ibselinux	3.4	3.4	
python3-l ibsemanage	3.4	3.4	
python3-l ibstoragemgmt	1.9.4		
python3-l ockfile	0.12.2		
python3-m arkupsafe	1.1.1	1.1.1	
python3-n etifaces	0.10.6	0.10.6	
python3-o authlib	3.0.2	3.0.2	
python3-pip- wheel	21.3.1	21.3.1	21.3.1
python3-ply	3.11	3.11	

パッケージ	AMI	Minimal AMI	コンテナ
python3-policycoreutils	3.4	3.4	
python3-pyrettytable	0.7.2	0.7.2	
python3-prompt-toolkit	3.0.24	3.0.24	
python3-pyycparser	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-setuptools	4.4.1	4.4.1	

パッケージ	AMI	Minimal AMI	コンテナ
python3-s etuptools	59.6.0	59.6.0	
python3-s etuptools- wheel	59.6.0	59.6.0	59.6.0
python3-six	1.15.0	1.15.0	
python3-systemd	235	235	
python3-urllib3	1.25.10	1.25.10	
python3-wcwidth	0.2.5	0.2.5	
python-chevron	0.13.1		
python-srpm- macros	3.9		
quota	4.06		
quota-nls	4.06		
readline	8.1	8.1	8.1
rng-tools	6.14	6.14	
rootfiles	8.1	8.1	
rpcbind	1.2.6		
rpm	4.16.1.3	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-libs	4.16.1.3	4.16.1.3	4.16.1.3

パッケージ	AMI	Minimal AMI	コンテナ
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	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		

パッケージ	AMI	Minimal AMI	コンテナ
strace	6.8		
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 年 3 月		
tcpdump	4.99.1		
tcsh	6.24.07		
time	1.9		
traceroute	2.1.3		

パッケージ	AMI	Minimal AMI	コンテナ
tzdata	2024a	2024a	2024a
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		
xfspgrog	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

パッケージ	AMI	Minimal AMI	コンテナ
yum	4.14.0	4.14.0	4.14.0
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	

コンテナでの AL2023 の使用

Note

AL2023 を使用して Amazon ECS でコンテナ化されたワークロードをホストする方法の詳細については、「[Amazon ECS コンテナホスト用 AL2023](#)」を参照してください。

ユースケースに応じて、コンテナ内で AL2023 を使用する方法はいくつかあります。[AL2023 ベースコンテナイメージ](#) は、Amazon Linux 2 コンテナイメージと AL2023 最小 AMI に最も似ています。

上級ユーザー向けに、AL2023.2 リリースで導入された最小コンテナイメージと、[ベアボーンコンテナ](#)の構築方法を説明するドキュメントを提供しています。

AL2023 は、AL2023 ベースのコンテナイメージ、または他の Linux 配布に基づくコンテナのいずれかの、コンテナ化されたワークロードをホストするためにも使用できます。[Amazon ECS コンテナホスト用 AL2023](#) を使用することも、付属のコンテナランタイムパッケージを直接使用することもできます。docker、containerd、nerdctl パッケージは AL2023 にインストールして使用できます。

トピック

- [AL2023 ベースコンテナイメージの使用](#)
- [AL2023 最小コンテナイメージ](#)

- [必要最低限の AL2023 コンテナイメージの構築](#)
- [Amazon Linux 2023 コンテナイメージにインストールされているパッケージの比較](#)
- [Amazon Linux 2023 最小 AMI とコンテナイメージにインストールされているパッケージの比較](#)

AL2023 ベースコンテナイメージの使用

AL2023 コンテナイメージは、AL2023 AMI に含まれているのと同じソフトウェアコンポーネントから構築されています。これは、Docker ワークロードのベースイメージとして任意の環境で使用できます。[Amazon Elastic Compute Cloud](#) (Amazon EC2) 内のアプリケーション用にすでに Amazon Linux AMI を使用している場合、Amazon Linux コンテナイメージでアプリケーションをコンテナ化できます。

ローカル開発環境で Amazon Linux コンテナイメージを使用し、Amazon [Elastic Container Service](#) (Amazon ECS) AWS を使用してアプリケーションをプッシュします。詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「[Amazon ECS で Amazon ECR イメージを使用する](#)」を参照してください。

Amazon Linux コンテナイメージは、Amazon ECR Public で入手できます。AL2023 に関するフィードバックは、指定された AWS 担当者を通じて、または GitHub の [amazon-linux-2023 リポジトリ](#) に問題を提出することで提供できます。

Amazon ECR Public から Amazon Linux コンテナイメージをプルする方法

1. Amazon Linux Public レジストリに Docker クライアントを認証します。認証トークンは 12 時間有効です。詳細については、「Amazon Elastic Container Registry ユーザーガイド」の「[プライベートルジストリの認証](#)」を参照してください。

Note

get-login-password コマンドは、最新バージョンの AWS CLI バージョン 2 を使用してサポートされています。詳細については、「AWS Command Line Interface ユーザーガイド」の「[Installing the AWS Command Line Interface](#)」を参照してください。

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

出力は次のとおりです。

```
Login succeeded
```

2. `docker pull` コマンドを起動して Amazon Linux コンテナイメージを取得します。Amazon ECR Public Gallery で Amazon Linux コンテナイメージを表示するには、「[Amazon ECR Public Gallery - amazonlinux](#)」を参照してください。

 Note

AL2023 Docker コンテナイメージを取得する場合、以下のいずれかの形式でタグを使用できます。

- AL2023 コンテナイメージの最新バージョンを取得するには、`:2023` タグを使用します。
- AL2023 の特定のバージョンを入手するには、以下の形式を使用できます。
 - `:2023.[0-7 release quarter].[release date].[build number]`

以下の例では、タグ `:2023` を使用して AL2023 の入手可能な最新のコンテナイメージを取得しています。

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

3. (オプション) コンテナをローカルに実行します。

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

Docker Hub から AL2023 コンテナイメージを取得する方法

1. `docker pull` コマンドを使用して AL2023 コンテナイメージを取得します。

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

2. (オプション) コンテナをローカルに実行します。

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

Note

AL2023 のコンテナイメージは、dnf パッケージマネージャーのみを使用してソフトウェアパッケージをインストールします。つまり、追加のソフトウェアに使用できる、amazon-linux-extras または同等のコマンドはありません。

AL2023 最小コンテナイメージ

Note

標準 AL2023 コンテナイメージは、ほとんどのユースケースに適しており、最小コンテナイメージへの導入は、通常の AL2023 コンテナイメージへの導入よりも作業が増える可能性があります。

AL2023.2 で導入された AL2023 最小コンテナイメージは、標準のベースコンテナイメージとは異なり、他のパッケージのインストールに必要な最低限のパッケージのみが含まれます。最小コンテナイメージは、便利なパッケージセットではなく、パッケージの最小限のセットとして設計されています。

AL2023 の最小コンテナイメージは、AL2023 ですでに利用可能なソフトウェアコンポーネントから構築されています。最小コンテナイメージの主な違いは、フル機能の Python ベースの dnf パッケージマネージャーではなく、microdnf を使用して dnf パッケージマネージャーを提供している点です。これにより、AL2023 AMI とベースコンテナイメージに含まれる dnf パッケージマネージャーの完全な機能セットが提供されないというトレードオフがあるものの、最小コンテナイメージのサイズは小さくなります。

AL2023 最小コンテナイメージは、provided.al2023 AWS Lambda ランタイム環境のベースを形成します。

最小コンテナイメージに含まれるパッケージの詳細なリストについては、「[Amazon Linux 2023 コンテナイメージにインストールされているパッケージの比較](#)」を参照してください。

最小コンテナイメージのサイズ

AL2023 の最小コンテナイメージに含まれるパッケージ数は、AL2023 ベースコンテナイメージよりも少ないため、サイズも大幅に小さくなります。Amazon Linux の現在と過去のリリースのコンテナイメージオプションの比較は、以下の表のとおりです。

Note

イメージサイズは [Amazon ECR Public Gallery の Amazon Linux](#) に表示されているとおりです。

イメージ	バージョン	イメージのサイズ	メモ
Amazon Linux 1 (AL1)	2018 年 3 月 0 日 2023 年 9 月 18 日	62.3 MB	x86-64 のみ
Amazon Linux 2	2.0.20230926.0	64.2 MB	aarch64 は x86-64 よりも 1.6 MB 大きくなります
Amazon Linux 2023 コンテナイメージ	2023 年 2 月 2 日 2023 年 1002.0	52.4 MB	
Amazon Linux 2023 最小コンテナイメージ	2023.2.20231002.0-minimal	35.2 MB	

AL2023 最小コンテナイメージの使用

AL2023 最小コンテナイメージは、ECR で利用可能で、2023-minimal タグは常に最新の AL2023 ベースの最小コンテナイメージを指しますが、minimal タグは AL2023 よりも新しいバージョンの Amazon Linux に更新される可能性があります。

これらのタグは、次の例のとおり、docker を使用してプルできます。

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

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

次の例は、最小コンテナイメージを取得し、その上に GCC をインストールする Dockerfile を説明しています。

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

必要最低限の AL2023 コンテナイメージの構築

AL2023 コンテナイメージは、AL2023 AMI に含まれているのと同じソフトウェアコンポーネントから構築されています。これには、パッケージマネージャー dnf など、ベースコンテナレイヤーを Amazon EC2 インスタンス上で実行する場合と同様に動作させるソフトウェアが含まれています。このセクションでは、アプリケーションに必要な最小限の依存関係のみを含むコンテナをゼロから構築する方法について説明します。

Note

標準 AL2023 コンテナイメージは、ほとんどのユースケースに適しています。標準のコンテナイメージを使用すると、イメージ上に簡単に構築できます。必要最低限のコンテナイメージの場合、既存のイメージ上に構築することがより困難になります。

アプリケーションの依存関係を最低限に抑えたコンテナを作成する方法

1. ランタイム依存関係を決定します。これはアプリケーションによって異なります。
2. FROM scratch をビルドする Dockerfile または Containerfile を作成します。次の Dockerfile の例では、bash シェルとその依存関係のみを含むコンテナを構築できます。

```
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"]
```

- この Dockerfile は以下のように機能します。

1. build という名前の AL2023 コンテナを起動します。このコンテナはベアボーンコンテナのブートストラップに使用されます。このコンテナ自体はデプロイされませんが、デプロイするコンテナが生成されます。
2. /sysroot ディレクトリを作成します。このディレクトリは、build コンテナがベアボーンコンテナに必要な依存関係をインストールする場所になります。以降のステップでは、/sysroot パスはベアボーンイメージのルートディレクトリとなるようにパッケージ化されます。

この `--installroot` オプションをこのように `dnf` に使用して、他の AL2023 イメージを作成します。これは、インストーラーとイメージ作成ツールを動作させる `dnf` の機能です。

3. `dnf` を呼び出して、/sysroot にパッケージをインストールします。

`rpm -q system-release --qf '%{VERSION}'` コマンドは `system-release` パッケージをクエリ (`-q`) し、クエリ対象のパッケージのバージョン (`%{VERSION}` 変数は RPM のバージョンに対応する `rpm` 変数) を出力するようにクエリ形式 (`--qf`) を設定します。

`dnf` の `--releasever` 引数を build コンテナ内の `system-release` バージョンに設定すると、Amazon Linux の更新されたコンテナベースイメージがリリースされるたびに、この Dockerfile を使用してベアボーンコンテナを再構築できます。

などの任意の Amazon Linux 2023 バージョン `--releasever` に を設定できます 2023.11.20260526。これにより、build コンテナは最新の AL2023 バージョンとして実行されますが、現在の AL2023 リリースに関係なく、2023.11.20260526 からベアボーンコンテナを構築します。

`--setopt=install_weak_deps=False` 設定オプションでは、推奨や提案ではなく、必要な依存関係のみをインストールするように `dnf` に指示します。

4. インストールしたシステムを空の (FROM scratch) コンテナのルートにコピーします。

5. この `/bin/bash` の場合は、`ENTRYPOINT` を目的のバイナリに設定します。

3. 空のディレクトリを作成し、ステップ 2 の例の内容を `Dockerfile` という名前のファイルに追加します。

```
$ 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. 次のコマンドを実行してコンテナを構築します。

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

5. 以下のコマンドを使用してコンテナを実行し、`bash` オンリーコンテナがどれほど最小限であるかを確認します。

```
$ 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
```

より実用的な例として、以下の手順では Hello World! を表示する C アプリケーション用のコンテナを構築します。

1. 空のディレクトリを作成し、C ソースコードおよび Dockerfile を追加します。

```
$ 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. 以下のコマンドを使用して、コンテナを構築します。

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

3. 以下のコマンドを実行して、コンテナを起動します。

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

Amazon Linux 2023 コンテナイメージにインストールされているパッケージの比較

AL2023 ベースコンテナイメージに存在する RPM と AL2023 最小コンテナイメージに存在する RPM の比較

パッケージ	コンテナ	最小コンテナ
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 年 2 月 68 日	2023 年 2 月 68 日
coreutils-single	8.32	8.32
crypto-policies	20220428	20220428
curl-minimal	8.5.0	8.5.0
dnf	4.14.0	

パッケージ	コンテナ	最小コンテナ
dnf-data	4.14.0	4.14.0
elfutils-default-yama-scope	0.188	
elfutils-libelf	0.188	
elfutils-libs	0.188	
expat	2.5.0	
file-libs	5.39	5.39
filesystem	3.14	3.14
gawk	5.1.0	5.1.0
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

パッケージ	コンテナ	最小コンテナ
json-c	0.14	0.14
keyutils-libs	1.6.3	1.6.3
krb5-libs	1.21.3	1.21.3
libacl	2.3.1	2.3.1
libarchive	3.7.4	3.7.4
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libblkid	2.37.4	2.37.4
libcap	2.48	2.48
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

パッケージ	コンテナ	最小コンテナ
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libnghttp2	1.59.0	1.59.0
libpeas		1.32.0
libpsl	0.21.1	0.21.1
librepo	1.14.5	1.14.5
libreport-fsfilesystem	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

パッケージ	コンテナ	最小コンテナ
libzstd	1.5.5	1.5.5
lua-libs	5.4.4	5.4.4
lz4-libs	1.9.4	1.9.4
microdnf		3.10.0
microdnf-dnf		3.10.0
mpfr	4.1.0	4.1.0
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
npth	1.6	1.6
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	

パッケージ	コンテナ	最小コンテナ
python3-libcomps	0.1.20	
python3-libdnf	0.69.0	
python3-libs	3.9.16	
python3-pip-wheel	21.3.1	
python3-rpm	4.16.1.3	
python3-setuptools-wheel	59.6.0	
readline	8.1	8.1
rpm	4.16.1.3	4.16.1.3
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

Amazon Linux 2023 最小 AMI とコンテナイメージにインストールされているパッケージの比較

AL2023 最小 AMI に存在する RPM と、AL2023 ベースコンテナイメージと最小コンテナイメージに存在する RPM の比較

パッケージ	Minimal AMI	コンテナ	最小コンテナ
alternatives	1.15	1.15	1.15
amazon-chrorny-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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
ca-certificates	2023 年 2 月 68 日	2023 年 2 月 68 日	2023 年 2 月 68 日
checkpolicy	3.4		
chrony	4.3		
cloud-init	22.2.2		
cloud-init-cfg-ec2	22.2.2		
cloud-utils-growpart	0.31		
coreutils	8.32		
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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
dbus-common	1.12.28		
dbus-libs	1.12.28		
device-mapper	1.02.185		
device-mapper-libs	1.02.185		
diffutils	3.8		
dnf	4.14.0	4.14.0	
dnf-data	4.14.0	4.14.0	4.14.0
dnf-plugin-n-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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
efi-filesystem	5		
efivar	38		
efivar-libs	38		
elfutils- default-yama- scope	0.188	0.188	
elfutils-libelf	0.188	0.188	
elfutils-libs	0.188	0.188	
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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
kernel-li vepatch-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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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-hmacalc	1.4.0		
libmnl	1.0.4		
libmodulemd	2.13.0	2.13.0	2.13.0
libmount	2.37.4	2.37.4	2.37.4
libnghttp2	1.59.0	1.59.0	1.59.0
libpeas			1.32.0
libpipeline	1.5.3		
libpsl	0.21.1	0.21.1	0.21.1

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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	

パッケージ	Minimal AMI	コンテナ	最小コンテナ
python3-dnf-plugins-core	4.3.0		
python3-d ocutils	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-j mespath	0.10.0		
python3-j sonpatch	1.21		
python3-j sonpointer	2.0		
python3-j sonschema	3.2.0		
python3-l ibcomps	0.1.20	0.1.20	
python3-libdnf	0.69.0	0.69.0	
python3-libs	3.9.16	3.9.16	
python3-l ibselinux	3.4		
python3-l ibsemanage	3.4		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
python3-m arkupsafe	1.1.1		
python3-n etifaces	0.10.6		
python3-o authlib	3.0.2		
python3-pip- wheel	21.3.1	21.3.1	
python3-ply	3.11		
python3-p olicycoreutils	3.4		
python3-p rettytable	0.7.2		
python3-prompt- toolkit	3.0.24		
python3-p ycparser	2.20		
python3-p yrsistent	0.17.3		
python3-p yserial	3.4		
python3-pysocks	1.7.1		
python3-pytz	2022.7.1		
python3-pyyaml	5.4.1		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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		

パッケージ	Minimal AMI	コンテナ	最小コンテナ
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		
xfspgrog	5.18.0		
xz	5.2.5		
xz-libs	5.2.5	5.2.5	5.2.5
yum	4.14.0	4.14.0	
zlib	1.2.11	1.2.11	1.2.11
zram-generator	1.1.2		
zram-generator-defaults	1.1.2		
zstd	1.5.5		

AWS Elastic Beanstalk での AL2023

AWS Elastic Beanstalk はウェブアプリケーションやサービスをデプロイ、スケールするためのサービスです。コードをアップロードするだけで、Elastic Beanstalk が、キャパシティーのプロビジョニング、ロードバランシング、自動スケーリングからアプリケーションの状態モニタリングまで、デプロイを自動的に処理します。詳細については、「[AWS Elastic Beanstalk](#)」を参照してください。

Elastic Beanstalk を使用するには、アプリケーションを作成し、アプリケーションソースバンドル (Java .war ファイルなど) の形式でアプリケーションバージョンを Elastic Beanstalk にアップロードした後、アプリケーションに関する情報を提供します。Elastic Beanstalk によって自動的に環境が起動され、コードの実行に必要な AWS リソースが作成および構成されます。詳細については、「[AWS Elastic Beanstalk デベロッパーガイド](#)」を参照してください。

Elastic Beanstalk Linux プラットフォームは Amazon EC2 インスタンスを使用し、これらのインスタンスは Amazon Linux を実行します。2023 年 8 月 4 日現在、Elastic Beanstalk は、Amazon Linux 2023 ベースの Docker、Tomcat、Java SE、Node.js、PHP、Python プラットフォームブランチを提供しています。Elastic Beanstalk は、AL2023 のサポートをより多くの Elastic Beanstalk プラットフォームにリリースする作業に取り組んでいます。

Elastic Beanstalk プラットフォームのサポートと AL2023 をベースに構築された現在のプラットフォームのすべてのリストは、「[Elastic Beanstalk デベロッパーガイド](#)」の「[Elastic Beanstalk Linux プラットフォーム](#)」セクションに記載されています。

新しい Elastic Beanstalk プラットフォームと既存のプラットフォームのバージョンのリリースノートは、「[Elastic Beanstalk リリースノート](#)」に記載されています。

AWS CloudShell での AL2023 の使用

AWS CloudShell はブラウザベースの事前に認証されたシェルで、 から直接起動できます。AWS マネジメントコンソールAWS マネジメントコンソール から CloudShell に移動するには、いくつかの方法があります。詳細については、「[AWS CloudShell の開始方法](#)」を参照してください。

AWS CloudShell は現在 Amazon Linux 2 に基づいていますが、今後 AL2023 に移行されます。AL2023 への移行は、2023 年 12 月 4 日からすべての AWS リージョン で開始されます。CloudShell の AL2023 への移行の詳細については、「[Amazon Linux 2 から Amazon Linux 2023 への AWS CloudShell の移行](#)」を参照してください。

AL2023 ベースの Amazon ECS AMI を使用したコンテナ化されたワークロードのホスト

Note

AL2023 をコンテナ内で使用する方法の詳細については、「[コンテナでの AL2023](#)」を参照してください。

Amazon Elastic Container Service (Amazon ECS) は、コンテナ化されたアプリケーションを簡単にデプロイ、管理、スケーリングできる、完全マネージド型のコンテナオーケストレーションサービスです。フルマネージドサービスである Amazon ECS には、AWS の設定と運用に関するベストプラクティスが組み込まれています。Amazon Elastic Container Registry (Amazon ECR) や Docker などの AWS とサードパーティーツールの両方に統合されています。この統合により、チームは環境ではなくアプリケーションの構築に集中しやすくなります。コントロールプレーンの複雑な管理は必要なく、クラウドの AWS リージョン間で、コンテナワークロードを実行およびスケールできます。

AL2023 ベースの Amazon ECS 最適化 AMI を使用して、コンテナ化されたワークロードを AL2023 でホストできます。詳細については、「[Amazon ECS 最適化 AMI](#)」を参照してください。

Amazon ECS 向けの AL2023 の変更 (AL2 との比較)

AL2 と同様に、AL2023 は Amazon ECS Linux インスタンスとして実行するために必要な基本パッケージを提供します。AL2 では、containerd、docker、および ecs-init パッケージを amazon-linux-extras から入手できましたが、AL2023 ではこれらのパッケージがコアリポジトリに含まれています。

バージョン管理されたリポジトリ機能を介した確定的なアップグレードにより、すべての AL2023 AMI はデフォルトで特定のリポジトリバージョンにロックされます。これは AL2023 Amazon ECS 最適化 AMI にも当てはまります。環境へのすべての更新をデプロイ前に慎重に管理およびテストできるだけでなく、問題が発生した場合に以前の AMI の内容に簡単に戻すことができます。AL2023 でのこの機能の詳細については、「[AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)」を参照してください。

AL2023 は AL2 でサポートされている cgroup v1 インターフェイスを介して cgroup v2 に切り替わります。詳細については、「[統合コントロールグループ階層 \(cgroup v2\)](#)」を参照してください。

 Note

[2023.2.20230920](#) (最初の AL2023.2 リリース) より前の AL2023 バージョンには、cgroup グループ内でのメモリ不足 (OOM) の処理に関して systemd にバグが含まれていました。cgroup 内のすべてのプロセスは常に強制終了となり、OOM-Killer が一度に 1 つのプロセスを選択するという本来の動作ではありませんでした。これは AL2 の動作と比較すると、リグレッションであり、AL2023 の 2023.2.20230920 リリースで修正されています。

Amazon ECS 最適化 AMI を構築するためのコードは、[amazon-ecs-ami GitHub プロジェクト](#)で入手できます。[リリースノート](#)では、どの AL2023 バージョンがどの Amazon ECS AMI バージョンにマッピングされるかについて説明しています。

AL2023 ベースの Amazon ECS 最適化 AMI のカスタマイズ

 Important

Amazon ECS 最適化 AL2023 AMI を使用することをお勧めします。詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「[Amazon ECS に最適化された AMI](#)」を参照してください。

Amazon ECS がカスタム AMI の作成に使用するのと同じビルドスクリプトを使用できます。詳細については、「[Amazon ECS 最適化 Linux AMI のビルドスクリプト](#)」を参照してください。

AL2023 での Amazon Elastic File System の使用

Amazon Elastic File System (Amazon EFS) は、サーバーレスで伸縮自在なファイルストレージを提供するため、ストレージ容量およびパフォーマンスのプロビジョニングや管理を行わずにファイルデータを共有できます。Amazon EFS は、アプリケーションを中断することなく、ファイルの追加や削除に伴って自動的に伸縮し、ペタバイト規模までオンデマンドで拡張できるように構築されています。Amazon EFS はシンプルなウェブサービスインターフェイスを提供しているため、ファイルシステムをすばやく簡単に作成、設定できます。このサービスでは、ユーザーに代わってすべてのファイルストレージインフラストラクチャを管理するため、複雑なデプロイ、パッチ適用、および複雑なファイルシステム設定の保守を行う必要がありません。

Amazon EFS はネットワークファイルシステムバージョン 4 (NFSv4.1 および NFSv4.0) プロトコルをサポートするので、現在お使いのアプリケーションやツールも Amazon EFS とシームレスに動作します。Amazon EC2、Amazon ECS、AWS Lambda などの、複数のコンピューティングインスタンスにより、1 つの Amazon EFS ファイルシステムに同時にアクセスできます。したがって、EFS ファイルシステムは、複数のコンピューティングインスタンスやサーバーで実行されるワークロードやアプリケーションに共通のデータソースを提供できます。

AL2023 への **amazon-efs-utils** のインストール

amazon-efs-utils パッケージは AL2023 リポジトリで入手でき、インストールして Amazon EFS ファイルシステムにアクセスするために使用できます。

amazon-efs-utils パッケージを AL2023 にインストールする

- 次のコマンドを使用して amazon-efs-utils をインストールします。

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

Amazon EFS ファイルシステムの AL2023 マウンティング

amazon-efs-utils をインストールすると、AL2023 インスタンスに Amazon EFS ファイルシステムを簡単にマウントできます。

Amazon EFS ファイルシステムを AL2023 にマウントする

- ファイルシステム ID を使ってマウントするには、次のコマンドを使用します。

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

また、TLS を使用して転送中のデータを暗号化するようにファイルシステムをマウントしたり、ファイルシステム ID の代わりに DNS 名またはマウントターゲット IP を使用したりできます。詳細については、「[EFS マウントヘルパーを使用した Amazon EC2 Linux インスタンスへのマウンティング](#)」を参照してください。

AL2023 上に構築された Amazon EMR の使用

Amazon EMR は、Apache Hadoop と AWS が提供するサービスを使用して、膨大な量のデータを効率良く簡単に処理できるウェブサービスです。

AL2023 ベースの Amazon EMR リリース

Amazon EMR リリース 7.0.0 は、AL2023 をベースに構築された最初のリリースです。このリリースでは、AL2023 が Amazon EMR の基本オペレーティングシステムであり、AL2023 のすべての利点を Amazon EMR で利用できます。詳細については、「[Amazon EMR 7.0.0 リリースノート](#)」を参照してください。

EKS での Amazon EMR

EKS 6.13 での Amazon EMR は、AL2023 をオプションとして導入した最初のリリースです。このリリースでは、Java 17 ランタイムとともに AL2023 をオペレーティングシステムとして Spark を起動できます。詳細については、「[Amazon EMR on EKS 6.13 リリースノート](#)」およびすべての「[Amazon EMR on EKS リリースノート](#)」を参照してください。

での AL2023 の使用 AWS Lambda

を使用すると AWS Lambda、サーバーのプロビジョニングや管理を行わずにコードを実行できます。使用したコンピューティング時間に対してのみお支払いいただきます。コードが実行中でなければ料金はかかりません。実質どのようなタイプのアプリケーションやバックエンドサービスでも、管理をまったく必要とせずにコードを実行できます。コードをアップロードするだけで、コードの実行とスケールに必要な処理はすべて Lambda により自動的に実行され、高い可用性が維持されます。

AL2023 の **provided.al2023** マネージドランタイムとコンテナイメージ

provided.al2023 ベースのランタイムは [AL2023 の最小コンテナイメージ](#) に基づいており、AL2023 ベースの Lambda マネージドランタイムと [コンテナのベースイメージ](#) を提供します。provided.al2023 ランタイムは AL2023 の最小コンテナイメージを基盤としているため、約 109 MB の provided.al2 ランタイムと比較して大幅に小さく、40 MB 未満となっています。

詳細については、「[Lambda ランタイム](#)」および「[Lambda コンテナイメージの使用](#)」を参照してください。

AL2023 ベースの Lambda ランタイム

Node.js 20、Python 3.12、Java 21、.NET 8 などのマネージド言語ランタイムの今後のリリースでは、AL2023 が基盤となり、[AL2023 ベースのランタイム の発表](#)で説明されているように `provided.al2023` がベースイメージとして使用されます。

AL2023 ベースの Lambda 関数

- [Go で記述された AL2023 Lambda 関数](#)
- [Rust で記述された AL2023 Lambda 関数](#)

詳細については、AWS Lambda デベロッパーガイドの「[Lambda ランタイム](#)」を参照してください。

チュートリアル

以降のチュートリアルでは、Amazon Linux 2023 (AL2023) を実行する Amazon EC2 インスタンスを使用して一般的なタスクを実行する方法を示します。動画チュートリアルについては、[AWS の講習動画とラボ](#)を参照してください。

AL2 の使用方法については、「Amazon EC2 ユーザーガイド」の「[Linux を実行する Amazon EC2 インスタンスのチュートリアル](#)」を参照してください。

チュートリアル

- [チュートリアル: AL2023 に LAMP サーバーをインストールする](#)
- [チュートリアル: AL2023 に SSL/TLS を設定する](#)
- [チュートリアル: AL2023 で WordPress ブログをホストする](#)
- [チュートリアル: AL2023 での Redis 6 から Valkey への移行](#)
- [チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする](#)
- [チュートリアル: AL2023 で TigerVNC サーバーを設定する](#)
- [AL2023 カーネルでの多世代 LRU \(MGLRU\) の使用](#)
- [チュートリアル: AL2023 に LibreOffice をインストールする](#)

チュートリアル: AL2023 に LAMP サーバーをインストールする

次の手順では、Apache ウェブサーバーを PHP と [MariaDB](#) (コミュニティによって開発された MySQL の派生版) のサポートとともに AL2023 インスタンスにインストールします (LAMP ウェブサーバーまたは LAMP スタックとも呼ばれます)。このサーバーを使用して静的ウェブサイトをホストしたり、データベースとの情報の読み取りと書き込みを行う動的な PHP アプリケーションをデプロイしたりできます。

Important

これらの手順は AL2023 で使用するためのものです。Ubuntu や Red Hat Enterprise Linux などの別のディストリビューションに LAMP ウェブサーバーを設定しようとする、このチュートリアルの通りにはなりません。Ubuntu については、Ubuntu コミュニティドキュメントの [ApacheMySQLPHP](#) を参照してください。その他のディストリビューションについては、それぞれのドキュメントを参照してください。

タスク

- [ステップ 1: LAMP サーバーを準備する](#)
- [ステップ 2: LAMP サーバーをテストする](#)
- [ステップ 3: データベースサーバーをセキュリティで保護する](#)
- [ステップ 4: \(オプション\) phpMyAdmin をインストールする](#)
- [トラブルシューティング](#)
- [関連トピック](#)

ステップ 1: LAMP サーバーを準備する

前提条件

- このチュートリアルでは、インターネットからアクセス可能なパブリック DNS 名を持つ、AL2023 を使用する新しいインスタンスをすでに起動していることを前提にしています。詳細については、「[Amazon EC2 での AL2023](#)」を参照してください。また、セキュリティグループを設定して、SSH (ポート 22)、HTTP (ポート 80)、HTTPS (ポート 443) 接続を有効にしている必要があります。前提条件の詳細については、「Amazon EC2 ユーザーガイド」の「[Linux インスタンスへのインバウンドトラフィックを許可する](#)」を参照してください。
- 次の手順により AL2023 で使用可能な最新の PHP バージョンがインストールされます。現在は 8.1 です。このチュートリアルで説明している以外の PHP アプリケーションを使用する場合は、8.1 との互換性を確認する必要があります。

LAMP サーバーを準備するには

1. インスタンスに接続します。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。
2. すべてのソフトウェアパッケージが最新の状態であることを確認するため、インスタンスでソフトウェアの更新を実行します。この処理には数分かかりますが、最新のセキュリティアップデートとバグ修正を必ず適用することが重要です。

-y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を検査する場合は、このオプションを省略できます。

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


3. Apache ウェブサーバーの最新バージョンと AL2023 用の PHP パッケージをインストールします。

```
[ec2-user ~]$ sudo dnf install -y httpd wget php-fpm php-mysqlcli php-json php php-devel
```

4. MariaDB ソフトウェアパッケージをインストールします。dnf install コマンドを使用すると、複数のソフトウェアパッケージと関連するすべての依存関係を同時にインストールできます。

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

次のコマンドを使用して、これらのパッケージの現在のバージョンを表示できます。

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

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

5. Apache ウェブサーバーを起動します。

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

6. systemctl コマンドを使用して、システムがブートするたびに Apache ウェブサーバーが起動するように設定します。

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

httpd が有効であることは、次のコマンドを実行して確認できます。

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

7. インバウンド HTTP (ポート 80) 接続をインスタンスに許可するセキュリティルールを追加していない場合には、このルールを追加します。デフォルトでは、起動時に launch-wizard-**N** セキュリティグループがインスタンス用に作成されます。その他のセキュリティグループルールを追加しなければ、このグループに含まれるのは SSH 接続を許可する単一のルールのみとなります。

- Amazon EC2 コンソールの <https://console.aws.amazon.com/ec2/> を開いてください。
- ナビゲーションペインで [インスタンス] を選択し、インスタンスを選択します。
- [セキュリティ] タブで、インバウンドルールを表示します。次のルールが表示されます。

Port range	Protocol	Source
22	tcp	0.0.0.0/0

Warning

0.0.0.0/0 を使用すると、すべての IPv4 アドレスからインスタンスへの、SSH によるアクセスが許可されます。これはテスト環境で短時間なら許容できますが、実稼働環境で行うのは安全ではありません。本番環境では、特定の IP アドレスまたは特定のアドレス範囲にのみ、インスタンスへのアクセスを限定します。

- HTTP (ポート 80) 接続を許可するインバウンドルールがないときは、この時点でルールを追加する必要があります。セキュリティグループのリンクを選択します。「[Linux インスタンス用のインバウンドトラフィック承認](#)」の手順を使用して、次の値で新しいインバウンドセキュリティルールを追加します。

- [Type]: HTTP

- [Protocol]: TCP
- [Port Range]: 80
- [Source]: Custom

8. ウェブサーバーをテストします。ウェブブラウザで、インスタンスのパブリック DNS アドレス (またはパブリック IP アドレス) を入力します。/var/www/html にコンテンツがない場合は Apache テストページが表示され、このページには「正常に動作しました」と表示されます。

インスタンスのパブリック DNS は、Amazon EC2 コンソールを使用して取得できます ([パブリック IPv4 DNS] 列を確認します。この列が表示されない場合は、[設定] (歯車のアイコン) をクリックし、[パブリック IPv4 DNS] を選択します)。

インスタンスのセキュリティグループに、ポート 80 での HTTP ラフィックを許可するルールが含まれていることを確認します。詳細については、「[セキュリティグループへのルールの追加](#)」を参照してください。

Important

Amazon Linux を使用していない場合は、それらの接続を許可するようにインスタンスのファイアウォールを設定する必要があるかもしれません。ファイアウォールの設定方法の詳細については、ディストリビューション用のドキュメントを参照してください。

Apache httpd は、Apache ドキュメントルートと呼ばれるディレクトリに維持されるファイルを提供します。Amazon Linux Apache ドキュメントルートは /var/www/html であり、デフォルトでは root によって所有されます。

ec2-user アカウントがこのディレクトリで複数のファイルを操作することを許可するには、ディレクトリの所有権とアクセス許可を変更する必要があります。このタスクを行うには、複数の方法があります。このチュートリアルでは、ec2-user を apache グループに追加し、/var/www ディレクトリの所有権を apache グループに付与し、グループへの書き込み権限を割り当てます。

ファイルの許可を設定するには

1. ユーザー (この場合は ec2-user) を apache グループに追加します。

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

2. ログアウトし、再度ログインして新しいグループを選択し、メンバーシップを確認します。

- a. ログアウトします (exit コマンドを使用するか、ターミナルウィンドウを閉じます)。

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

- b. apache グループのメンバーシップを検証するには、インスタンスに再接続して次のコマンドを実行します。

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

3. /var/www とそのコンテンツのグループ所有権を apache グループに変更します。

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

4. グループの書き込み許可を追加して、これからのサブディレクトにグループ ID を設定するには、/var/www とサブディレクトのディレクトリ許可を変更します。

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

5. グループ書き込み許可を追加するには、/var/www とサブディレクトリのファイル許可を再帰的に変更します。

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

ここで、ec2-user (および apache グループの将来のメンバー) は、Apache ドキュメントルートでファイルを追加、削除、編集できるようになります。したがって、静的ウェブサイトや PHP アプリケーションなどのコンテンツを追加できます。

ウェブサーバーを保護するには (オプション)

HTTP プロトコルを実行するウェブサーバーは、送受信したデータのトランスポートセキュリティを提供しません。ウェブブラウザを使用して HTTP サーバーに接続すると、閲覧した URL、受信したウェブページのコンテンツ、送信した HTML フォームの内容 (パスワードなど) はすべて、ネットワーク経路上のだれでも傍受できるようになります。ウェブサーバーを保護するためのベストプラクティスとして、SSL/TLS 暗号化でデータを保護する HTTPS (HTTP Secure) のサポートをインストールしてください。

サーバーで HTTPS を有効にする方法については、「[チュートリアル: AL2023 に SSL/TLS を設定する](#)」を参照してください。

ステップ 2: LAMP サーバーをテストする

サーバーがインストールおよび実行されており、ファイルのアクセス許可が正しく設定されている場合、ec2-user アカウントは、インターネットから使用できる /var/www/html ディレクトリに PHP ファイルを作成できます。

LAMP サーバーをテストするには

1. Apache ドキュメントルートで PHP ファイルを作成します。

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

このコマンドを実行しようとしたときに「許可が拒否されました」というエラーが表示された場合は、ログアウトし、再度ログインして、[ファイルの許可を設定するには](#) で設定した正しいグループ許可を取得します。

2. ウェブブラウザで、作成したファイルの URL を入力します。この URL は、インスタンスのパブリック DNS アドレスにスラッシュとファイル名を追加したものです。次に例を示します。

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

PHP 情報ページが表示されるはずです。

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-xmldr.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

このページが表示されない場合は、前のステップで `/var/www/html/phpinfo.php` ファイルが正しく作成されたことを確認します。次のコマンドで、必要なパッケージがすべてインストールされたことを確認することもできます。

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

必要なパッケージのいずれかが出力に表示されていない場合は、`sudo yum install package` コマンドを使ってインストールします。

3. `phpinfo.php` ファイルを削除します。これは有用な情報であることもありますが、セキュリティ上の理由から、インターネット上で公表しないでください。

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

これで、完全に機能する LAMP ウェブサーバーを設定しました。`/var/www/html` の Apache ドキュメントルートにコンテンツを追加する場合、そのコンテンツはインスタンスのパブリック DNS アドレスで表示できます。

ステップ 3: データベースサーバーをセキュリティで保護する

MariaDB サーバーのデフォルトのインストールには、テストおよび開発に役立ついくつかの機能がありますが、実稼働サーバーでは無効にするか削除する必要があります。`mysql_secure_installation` コマンドを使用すると、ルートパスワードを設定し、安全でない機能をインストールから削除する手順が案内されます。MariaDB サーバーを使用する予定がない場合でも、この手順を実行することが推奨されます。

MariaDB サーバーをセキュリティで保護するには

1. MariaDB サーバーを起動します。

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

2. `mysql_secure_installation` を実行します。

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

- a. プロンプトが表示されたら、ルートアカウントのパスワードを入力します。
 - i. 現在のルートパスワードを入力します。デフォルトでは、ルートアカウントにはパスワードが設定されていません。Enter キーを押します。
 - ii. 「Y」と入力してパスワードを設定し、安全なパスワードを 2 回入力します。安全なパスワード作成の詳細については、「<https://identitysafe.norton.com/password-generator/>」を参照してください。このパスワードは必ず安全な場所に保管します。

MariaDB のルートパスワードの設定は、データベースを保護するための最も基本的な手段にすぎません。データベース駆動型アプリケーションを構築またはインストールす

る必要がある場合、通常はそのアプリケーションのデータベースサービスユーザーを作成します。ルートアカウントは、データベース管理以外には使用しないでください。

- b. 「Y」と入力して匿名ユーザーアカウントを削除します。
 - c. 「Y」と入力してリモートルートログインを無効にします。
 - d. 「Y」と入力してテストデータベースを削除します。
 - e. 「Y」と入力して権限テーブルを再ロードし、変更を保存します。
3. (オプション) MariaDB サーバーをすぐに使用する予定がない場合は、これを停止します。再び必要になったときには再起動できます。

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

4. (オプション) ブート時に毎回 MariaDB サーバーを起動させる場合は、次のコマンドを入力します。

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

ステップ 4: (オプション) phpMyAdmin をインストールする

[phpMyAdmin](#) は、EC2 インスタンスで MySQL データベースを表示して編集するために使用できる、ウェブベースのデータベース管理ツールです。Amazon Linux インスタンスで phpMyAdmin をインストールして設定するには、以下の手順に従ってください。

Important

Apache で SSL/TLS を有効にしていない場合、LAMP サーバーへのアクセスに phpMyAdmin を使用することは推奨されません。そのようにすると、データベース管理者のパスワードや他のデータは、インターネット上を安全ではない状態で送信されます。開発者によるセキュリティ関連の推奨事項については、「[Securing your phpMyAdmin installation](#)」を参照してください。EC2 インスタンスでのウェブサーバーの保護に関する一般的な情報については、「[チュートリアル: AL2023 に SSL/TLS を設定する](#)」を参照してください。

phpMyAdmin をインストールするには

1. 必要な依存ファイルをインストールします。


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

2. Apache を再起動します。

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

3. php-fpm を再起動します。

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

4. /var/www/html で Apache ドキュメントルートに移動します。

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

5. <https://www.phpmyadmin.net/downloads> で最新の phpMyAdmin リリース用のソースパッケージを選択します。ファイルディレクトリをインスタンスにダウンロードするには、次の例のようにリンクをコピーして wget コマンドに貼り付けます。

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

6. phpMyAdmin フォルダを作成し、次のコマンドでパッケージを展開します。

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

7. *phpMyAdmin-latest-all-languages.tar.gz* Tarball を削除します。

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

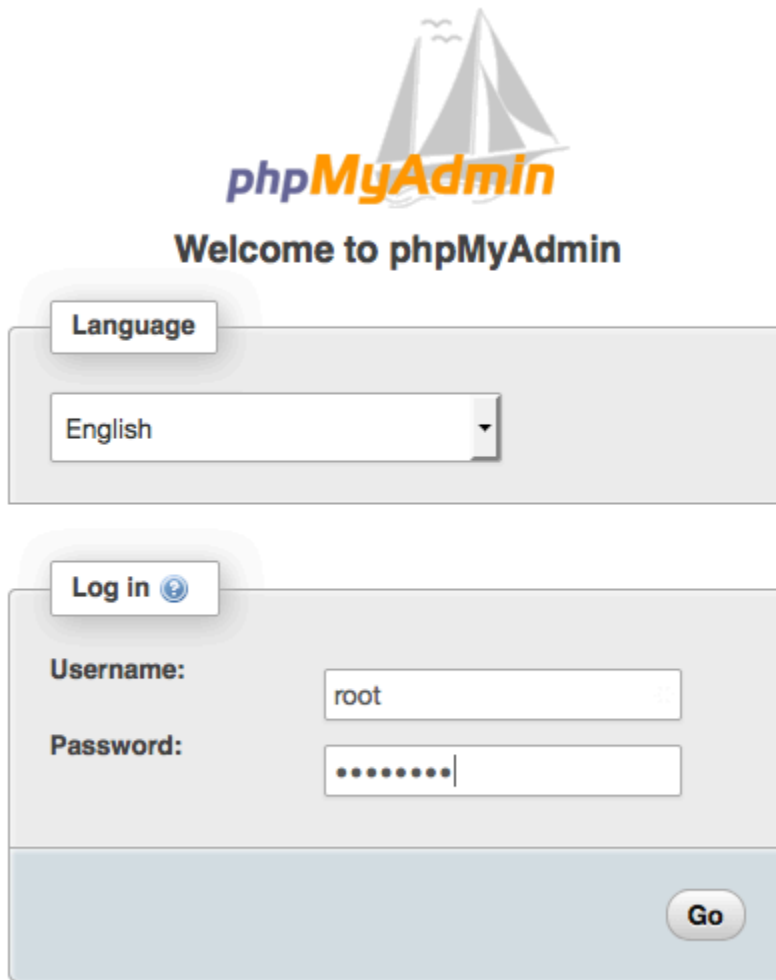
8. (オプション) MySQL サーバーが実行中ではない場合は、今すぐ起動します。

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

9. ウェブブラウザで、phpMyAdmin のインストール URL を入力します。この URL は、インスタンスのパブリック DNS アドレス (または、パブリック IP アドレス) にスラッシュとインストールディレクトリを追加してものです。次に例を示します。

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

phpMyAdmin ログインページが表示されます。



10. 前に作成した root ユーザー名と MySQL のルートパスワードを使って、phpMyAdmin インストールにログインします。

インストールは、サービス開始前に設定する必要があります。次の手順に従って、設定ファイルを手動で作成することから始めるのをお勧めします。

- a. 最小の設定ファイルから開始するには、お気に入りのテキストエディタを使用して新しいファイルを作成し、config.sample.inc.php の内容をそのファイルにコピーします。
- b. index.php を含む phpMyAdmin ディレクトリに、ファイルを config.inc.php として保存します。
- c. 追加のセットアップについては、phpMyAdmin のインストール手順の「[セットアップスクリプトの使用](#)」セクションにある「ファイル作成後の手順」を参照してください。

phpMyAdmin の使用に関する情報は、「[phpMyAdmin ユーザーガイド](#)」を参照してください。

トラブルシューティング

このセクションでは、新しい LAMP サーバーの設定時に発生する可能性がある一般的な問題の解決案を提供します。

ウェブブラウザを使用してサーバーに接続できません。

以下のチェックを行って、Apache ウェブサーバーが実行されていて、アクセス可能であるかどうかを確認します。

- ウェブサーバーが実行されていますか？

httpd が有効であることは、次のコマンドを実行して確認できます。

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

httpd プロセスが実行されていない場合は、[LAMP サーバーを準備するには](#) に記載されているステップを繰り返します。

- ファイアウォールは正しく設定されていますか？

インスタンスのセキュリティグループに、ポート 80 での HTTP ラフィックを許可するルールが含まれていることを確認します。詳細については、「[セキュリティグループへのルールの追加](#)」を参照してください。

HTTPS を使用してサーバーに接続できない

以下のチェックを行って、Apache ウェブサーバーが HTTPS をサポートするように設定されているかどうかを確認します。

- ウェブサーバは正しく設定されていますか？

Apache をインストールすると、サーバーは HTTP トラフィック用に設定されます。HTTPS をサポートするには、サーバーで TLS を有効にし、SSL 証明書をインストールします。詳細については、「[チュートリアル: AL2023 に SSL/TLS を設定する](#)」を参照してください。

- ファイアウォールは正しく設定されていますか？

インスタンスのセキュリティグループに、ポート 443 で HTTPS トラフィックを許可するルールが含まれていることを確認します。詳細については、の「[インスタンス用のインバウンドトラフィックの承認](#)」を参照してください。

関連トピック

インスタンスへのファイルの転送、またはウェブサーバーへの WordPress ブログのインストールの詳細については、次のドキュメントを参照してください。

- 「Amazon EC2 ユーザーガイド」の「[WinSCP を使用して Linux インスタンスにファイルを転送する](#)」。
- Amazon EC2 ユーザーガイド」の「[SCP クライアントを使用して Linux インスタンスにファイルを転送する](#)」。
- [チュートリアル: AL2023 で WordPress ブログをホストする](#)

このチュートリアルで使用されているコマンドおよびソフトウェアの詳細については、次のウェブページを参照してください。

- Apache ウェブサーバー: <http://httpd.apache.org/>
- MariaDB データベースサーバー: <https://mariadb.org/>
- PHP プログラミング言語: <http://php.net/>

ウェブサーバーのドメイン名の登録、または、既存のドメイン名をこのホストに移す方法についての詳細は、『Amazon Route 53 デベロッパーガイド』の「[Amazon Route 53 のドメインとサブドメインの作成と移行](#)」を参照してください。

チュートリアル: AL2023 に SSL/TLS を設定する

Secure Sockets Layer/Transport Layer Security (SSL/TLS) は、ウェブサーバーとウェブクライアントの間に、転送中のデータが傍受されないように保護する、暗号化されたチャネルを確立します。このチュートリアルでは、AL2023 と Apache ウェブサーバーを使用して、EC2 インスタンスに SSL/TLS のサポートを手動で追加する方法について説明します。このチュートリアルでは、ロードバランサーを使用していないことを前提としています。Elastic Load Balancing を使用している場合は、代わりに [AWS Certificate Manager](#) の証明書を使用して、ロードバランサーで SSL オフロードを設定できます。

歴史的経緯から、ウェブの暗号化は、単純に SSL と呼ばれることが少なくありません。ウェブブラウザでは今でも SSL がサポートされていますが、後継プロトコルである TLS プロトコルの方が攻撃を受けにくくなります。AL2023 では、すべてのバージョンの SSL でデフォルトでサーバー側のサポートが無効に設定されています。[セキュリティ標準化団体](#)は、TLS 1.0 は安全でないとみなしています。TLS 1.0 および TLS 1.1 は、2021 年 3 月に正式に[非推奨になりました](#)。このチュートリアルは、TLS 1.2 を有効にすることを前提としたガイダンスです。TLS 1.3 は 2018 年に最終化され、基盤となる TLS ライブラリ (このチュートリアルでは OpenSSL) がサポートされているため、有効に設定されている限り、AL2 で利用できます。[クライアントは 2023 年 6 月 28 日までに TLS 1.2 以降をサポートしている必要があります](#)。最新の暗号化基準の詳細については、「[RFC 7568](#)」および「[RFC 8446](#)」を参照してください。

このチュートリアルでは、現代のウェブ暗号化を単に TLS と呼びます。

Important

これらの手順は AL2023 で使用するためのものです。異なるディストリビューションを実行している EC2 インスタンス、または古いバージョンの Amazon Linux を実行しているインスタンスをセットアップしようとする、このチュートリアルの一部の手順が上手くいかないことがあります。Ubuntu については、[Ubuntu 上の OpenSSL](#) に関する Ubuntu コミュニティドキュメントを参照してください。Red Hat Enterprise Linux については、以下を参照してください。[Apache HTTP Web サーバーの設定](#)。その他のディストリビューションについては、それぞれのドキュメントを参照してください。

Note

または、AWS Nitro Enclaves に AWS Certificate Manager (ACM) を使用することもできます。これは、AWS Nitro Enclaves で Amazon EC2 インスタンスで実行されているウェブアプリケーションとサーバーでパブリックおよびプライベートの SSL/TLS 証明書を使用できるようにするエンクレーブアプリケーションです。Nitro Enclaves は、SSL/TLS 証明書やプライベートキーなどの機密性の高いデータを保護し、安全に処理するために、分離されたコンピューティング環境を作成できる Amazon EC2 の機能です。

Nitro Enclaves 向け ACM では、Amazon EC2 Linux インスタンスで実行する nginx を使用することで、プライベートキーの作成、証明書とプライベートキーの配布、および証明書の更新を実行します。

Nitro Enclaves 向け ACM を使用するには、エンクレーブ対応の Linux インスタンスを使用する必要があります。

詳細については、[AWS Nitro Enclaves ユーザーガイドの「Nitro Enclaves とはAWS AWS Certificate Manager」](#) および [「Nitro Enclaves 用」](#) を参照してください。

内容

- [前提条件](#)
- [ステップ 1: サーバーで TLS を有効にする](#)
- [ステップ 2: CA 署名証明書を取得する](#)
- [ステップ 3: セキュリティ設定をテストして強化する](#)
- [トラブルシューティング](#)

前提条件

このチュートリアルを開始する前に、次のステップを完了してください。

- EBS ベースの AL2023 インスタンスを起動します。詳細については、「[Amazon EC2 での AL2023](#)」を参照してください。
- インスタンスが以下の TCP ポートで接続を受け付けるようにセキュリティグループを設定します。
 - SSH (ポート 22)
 - HTTP (ポート 80)
 - HTTPS (ポート 443)

詳細については、「Amazon EC2 ユーザーガイド」の「[Authorize inbound traffic for your Linux instances](#)」を参照してください。

- Apache ウェブサーバーをインストールします。手順については、「[チュートリアル: AL2023 に LAMP サーバーをインストールする](#)」を参照してください。必要なのは httpd パッケージおよび対応する従属コンポーネントのみです。PHP および MariaDB に関連する手順は無視してかまいません。
- ウェブサイトの識別と認証を行うため、TLS の公開鍵基盤 (PKI) ではドメインネームシステム (DNS) を使用します。EC2 インスタンスを使用してパブリックウェブサイトをホストするには、ウェブサーバーのドメイン名を登録するか、既存のドメイン名を Amazon EC2 ホストに移す必要があります。これについては、ドメイン登録および DNS ホスティングに関するサードパーティのサービスが多数存在します。[Amazon Route 53](#) を使用することもできます。

ステップ 1: サーバーで TLS を有効にする

この手順では、自己署名のデジタル証明書を使用して、AL2023 で TLS を設定します。

Note

自己署名証明書はテスト用であり、本稼働環境では使用できません。インターネットに自己署名証明書を公開すると、サイトへの訪問者にセキュリティ警告が表示されます。

サーバーで TLS を有効にするには

1. インスタンスに接続し、Apache が実行されていることを確認します。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。

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

返される値が「enabled」でない場合、Apache を起動し、システムブート時に毎回起動されるように設定します。

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

2. すべてのソフトウェアパッケージが最新の状態であることを確認するため、インスタンスでソフトウェアの更新を実行します。この処理には数分かかりますが、最新の更新とバグ修正を確実に適用することが重要です。

Note

-y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前に更新を検査する場合は、このオプションを省略できます。

```
[ec2-user ~]$ sudo dnf install openssl mod_ssl
```

3. 次のコマンドを入力すると、サイトに関する情報を入力できるプロンプトが表示されます。

```
[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
```

これにより、次の 2 つのファイルが生成されます。

- /etc/pki/tls/private/apache-selfsigned.key (プライベートキー)
- /etc/pki/tls/certs/apache-selfsigned.crt (自己署名証明書)

証明書ファイル名は、の SSLCertificateFile ディレクティブで割り当てられているデフォルトと一致します/etc/httpd/conf.d/ssl.conf。

次のファイルがインスタンスに作成されました。このファイルは、セキュアサーバーの設定とテスト用の証明書の作成に使用します。

- /etc/httpd/conf.d/ssl.conf

mod_ssl の設定ファイル。このファイルには、暗号化キーと証明書の場所、許可する TLS プロトコル、受け入れる暗号化アルゴリズムを Apache に指示するディレクティブが含まれています。これはローカル証明書ファイルになります。

- /etc/pki/tls/certs/apache-selfsigned.crt
- /etc/pki/tls/private/apache-selfsigned.key

.crt ファイルには自己署名証明書が含まれ、.key ファイルにはプライベートキーが含まれています。Apache では、証明書とキーを PEM 形式にする必要があります。これは、次の短縮化された例のように、"BEGIN" 行と "END" 行で囲まれた Base64 エンコードの ASCII 文字で構成されます。

```
-----BEGIN PRIVATE KEY-----
MIIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQD2KKx/8Zk94m1q
3gQMZF9ZN66Ls19+3tHAgQ5Fpo9KJDhzLj00CI8u1PTcGmAah5kEitCEc0wzmNeo
BC10wYR6G0rGaKtK9Dn7CuIjvubtUysVyQoMVPQ97ldeakHWeRMiEJFXg6kZZ0vr
GvwnKoMh3D1K44D9dX7IDua2P1Yx5+eroA+1Lqf32ZSaA00bBIMIYTHigwbHMZoT
...
56tE7THvH7v0Ef4/iU0sIrEzaMaJ0mqkmY1A70qQGQKBgBF3H1qNRNHuyMcP0DFs
27hDzPDinrquSEvoZlggkDMLh2irTiipJ/GhkvTpoQlv0fK/VXw8vSgeaBuhwJvS
LXU9HvYq0U604FgD3nAyB9hI0BE13r1HjUvbjT7moH+RhnNz6eqqdscCS09VtRAo
```



```
4QQvAq0a8UheYeoXLdWcHaLP
-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----
MIIEAzCCA10gAwIBAgICWxQwDQYJKoZIhvcNAQELBQAwgbExCzAJBgNVBAYTAi0t
MRIwEAYDVQQIDAlTb211U3RhdGUxETAPBgNVBACMFNvbWVDaXR5MRkwFwYDVQQK
DBBTb211T3JnYW5pemF0aW9uMR8wHQYDVQQLDBZTb211T3JnYW5pemF0aW9uYXV
bm10MRkwFwYDVQQDDDBBpcC0xNzItMzEtMjAtMjMMSQwIgYJKoZIhvcNAQkBFhVy
...
z5rRUE/XzxRLBZ0oWZpNWTXJkQ3uFYH6s/sBwtHpKKZMz0vDedREjNKAvk4ws6F0
CuIjvubtUysVyQoMVPQ971deakHWeRMiEJFXg6kZZ0vrGvwnKoMh3D1K44D9d1U3
WanXWehT6FiSZvB4sTEXXJN2jdw8g+sHGnZ8zC0sclknYhHrCVD2vnBlZJKSZvak
3ZazhBxtQSukFM0nWPP2a0DMMFGYUHOd0BQE8sBJxg==
-----END CERTIFICATE-----
```

ファイル名および拡張子は利便性のためであり、機能には影響しません。例えば、`cert.crt` または `cert.pem` などのファイル名で証明書と呼び出すことができます。ただし、`ssl.conf` ファイルの関連ディレクティブが同じ名前を使用している場合に限りです。

Note

デフォルトの TLS ファイルを独自にカスタマイズしたファイルに置き換える場合は、PEM 形式であることを確認してください。

4. Apache を再起動します。

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

Note

前述のとおり、TCP 443 番ポートが EC2 インスタンスでアクセス可能であることを確認してください。

5. Apache ウェブサーバーではポート 443 経由で HTTPS (セキュア HTTP) がサポートされるようになっています。これをテストするには、ブラウザの URL バーに、**https://** というプレフィックスを指定して、EC2 インスタンスの IP アドレスまたは完全修飾ドメイン名を入力します。

信頼されていない自己署名ホスト証明書を使用してサイトに接続しようとしているため、ブラウザには一連のセキュリティ警告が表示されることがあります。この警告を無視し、サイトに進みます。

サーバーで TLS を正しく設定できていれば、Apache のデフォルトのテストページが開きます。これで、ブラウザとサーバーの間にやり取りされるすべてのデータが暗号化されるようになります。

Note

サイト訪問者に対して警告画面が表示されないようにするには、暗号化だけでなく、サイト所有者のパブリック認証を行うための信頼された CA 署名証明書を取得する必要があります。

ステップ 2: CA 署名証明書を取得する

CA 署名証明書を取得するには、次の手順に従います。

- プライベートキーから証明書署名リクエスト (CSR) を作成します。
- 作成した CSR を認証機関 (CA) に送信します。
- 署名付きホスト証明書を入手する
- 証明書を使用するように Apache を設定します

自己署名 TLS X.509 ホスト証明書は、暗号化技術上は CA 署名証明書と同じです。これらの相違は数学的なものではなく、社会的なものです。CA では、最低でもドメイン所有権を検証してから申請者に証明書を発行することを保証しています。そのため、各ウェブブラウザには、ブラウザベンダーが信頼する CA のリストが含まれています。X.509 証明書は主に、プライベートサーバーキーに対応するパブリックキーと、このパブリックキーに暗号で関連付けられている CA による署名で構成されています。HTTPS 経由でブラウザがウェブサーバーに接続すると、サーバーは、信頼された CA のリストをブラウザが確認できるように、証明書を提示します。Signer がリストに含まれている場合や、他の信頼された署名者の信頼チェーンを通じてアクセス可能である場合、ブラウザはサーバーと、高速暗号化データチャネルのネゴシエーションを行い、ページをロードします。

証明書には、リクエストの確認作業が必要であり、一般的に費用がかかるため、各社を比較することをお勧めします。いくつかの CA では、基本レベル証明書が無料で提供されます。これらの CA で最

も注目すべきは [Let's Encrypt](#) プロジェクトです。このプロジェクトでは、証明書の作成および更新プロセスの自動化もサポートしています。Let's Encrypt 証明書の使用の詳細については、「[Certbot の取得](#)」を参照してください。

商業グレードのサービスを提供する予定がある場合は、[AWS Certificate Manager](#) は良い選択肢です。

ホスト証明書の基盤にはキーがあります。2019 年時点で、[政府](#)および[業界グループ](#)は、2030 年まで、ドキュメントを保護するための RSA キーに 2048 ビットの最小キー (モジュロ) サイズを使用することを推奨しています。AL2023 で OpenSSL が生成するデフォルトのモジュラスサイズは 2048 ビットであるため、CA 署名証明書に適しています。次の手順では、モジュラスサイズを大きくする、別の暗号化アルゴリズムを使用するなど、キーのカスタマイズが必要な場合のオプションのステップを提供しています。

 Important

CA 署名ホスト証明書を取得するための手順は、登録およびホスト済みの DNS ドメインを所有している場合を除き、使用しません。

CA 署名証明書を取得するには

1. インスタンスに接続して、`/etc/pki/tls/private/` に移動します。サーバーの TLS 用プライベートキーは、このディレクトリに格納されます。既存のホストキーを使用して CSR を生成する場合は、ステップ 3 に進みます。インスタンスへの接続の詳細については「[AL2023 インスタンスへの接続](#)」を参照してください。
2. (オプション) 新しいプライベートキーを生成します。キー設定のいくつかのサンプルを次に示します。生成されたキーのどれもウェブサーバーで機能しますが、実装されるセキュリティの強度とタイプはそれぞれ異なります。

- 例 1: デフォルトの RSA ホストキーを作成します。結果として生成されるファイル **custom.key** が、2048 ビットの RSA プライベートキーです。

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key
```

- 例 2: これより大きなモジュラスサイズを使用して、より強力な RSA キーを作成します。結果として生成されるファイル **custom.key** が、4096 ビットの RSA プライベートキーです。

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key 4096
```

- 例 3: パスワードで保護された 4096 ビット暗号化 RSA キーを作成します。結果のファイル、**custom.key** は、AES-128 暗号で暗号化された 4096 ビットの RSA プライベートキーです。

 Important

キーを暗号化するとセキュリティを強化できますが、暗号化キーにはパスワードが必要であるため、暗号化に依存するサービスを自動的に開始することはできません。このキーを使用するたびに、SSH 接続でパスワード (前述の例では、"abcde12345") を指定する必要があります。

```
[ec2-user ~]$ sudo openssl genrsa -aes128 -passout pass:abcde12345 -out  
custom.key 4096
```

- 例 4: 非 RSA 暗号を使用してキーを作成します。RSA 暗号化は、2 つの大きな素数の積に基づくパブリックキーのサイズのために、比較的遅くなる可能性があります。ただし、非 RSA 暗号化方式を使用する TLS 用のキーを作成することも可能です。同等レベルのセキュリティを提供する場合は、楕円曲線の計算に基づいたキーのほうが小さく計算処理も高速です。

```
[ec2-user ~]$ sudo openssl ecparam -name prime256v1 -out custom.key -genkey
```

結果は、prime256v1 (OpenSSL でサポートされる "名前付き曲線") を使用した 256 ビットの楕円曲線プライベートキーです。暗号化強度は ([NIST](#) によると) 2048 ビットの RSA キーよりやや優れています。

 Note

すべての CA で、楕円曲線ベースのキーに対して RSA キーと同じレベルのサポートが提供されているわけではありません。

新しいプライベートキーには、制限の厳しい所有権とアクセス権を設定します (所有者 = root、グループ = root、所有者のみの読み取り/書き込み)。コマンドは次の例のようになります。

```
[ec2-user ~]$ sudo chown root:root custom.key  
[ec2-user ~]$ sudo chmod 600 custom.key
```

```
[ec2-user ~]$ ls -al custom.key
```

上記のコマンドにより、次のような結果が得られます。

```
-rw----- root root custom.key
```

適切なキーを作成し、設定できたら、CSR を作成できます。

3. 好みのキーを使用して CSR を作成します。次の例では **custom.key** を使用しています。

```
[ec2-user ~]$ sudo openssl req -new -key custom.key -out csr.pem
```

OpenSSL によりダイアログが開かれ、次の表に示されている情報の入力が必要です。基本的なドメイン検証済みホスト証明書については、[共通名] 以外のフィールドはすべてオプションです。

名前	説明	例
国名	2 文字の ISO 略称 (国名コード)。	US (= 米国)
州名	あなたが所属する組織の所在地の州または県。省略不可です。	ワシントン
市区町村	市など、組織の場所。	シアトル
組織名	組織の正式名称。組織名は、省略不可です。	Example Corp
部門名	組織に関する追加情報 (存在する場合)。	Example Dept
共通名	この値は、ユーザーがブラウザに入力する必要のあるウェブアドレスと正確に一致します。通常、これはプレフィックス付きのホスト名またはエイリアスによるドメイン名 (www.example.com の形式) を意味します。自己署名証明書を使用し、DNS 解決なしでテストを行う場合、共通名の構成要素はホスト名のみになる場合があります。CA では、 *.example.com などのワイルドカー	www.example.com

名前	説明	例
	ド名を許容する、よりコストの高い証明書も用意されています。	
E メールアドレス	サーバー管理者の E メールアドレス。	someone@example.com

最後に、OpenSSL により、オプションのチャレンジパスワードが求められます。このパスワードは CSR と、ユーザーと CA の間のトランザクションのみに適用されるため、このフィールドと、もう 1 つのオプションフィールドである、オプションの会社名については、CA の推奨事項に従ってください。CSR のチャレンジパスワードは、サーバー操作には影響しません。

結果として生成されるファイル **csr.pem** には、パブリックキー、パブリックキーのデジタル署名、入札したメタデータが含まれています。

- CA に CSR を送信します。この作業は通常、テキストエディタで CSR ファイルを開く動作と、内容をウェブフォームにコピーする動作で構成されています。このとき、証明書に適用する 1 つ以上のサブジェクト代替名 (SAN) を指定するように求められることがあります。共通名が **www.example.com** の場合、有効な SAN は **example.com** になります (逆も同様です)。サイトへの訪問者がこれら名前のいずれかを入力すると、エラーなしの接続が提示されます。CA のウェブフォームで許可される場合は、SAN のリストに共通名を含めます 一部の CA では自動的に含められます。

リクエストが承認されると、CA によって署名された新しいホスト証明書が届きます。CA の信頼チェーンを完成するために必要な、追加の証明書が含まれている中間証明書ファイルをダウンロードするよう指示されることもあります。

Note

多様な用途向けに複数の形式のファイルを送信してくる CA もあります。このチュートリアルでは、PEM 形式の証明書ファイルのみ使用してください。PEM 形式のファイルには通常、**.pem** または **.crt** ファイル拡張子が使用されます (ただし、常にこれらの拡張子が使用されるわけではありません)。どのファイルを使用すべきかわからない場合は、テキストエディタでファイルを開き、以下の行で始まる 1 つ以上のブロックを含むファイルを見つけてください。

```
- - - - -BEGIN CERTIFICATE - - - - -
```

ファイルの末尾は次のような行になっている必要があります。

```
- - - -END CERTIFICATE - - - - -
```

以下に示すように、コマンドラインでファイルをテストすることもできます。

```
[ec2-user certs]$ openssl x509 -in certificate.crt -text
```

これらの行がファイルに表示されていることを確認してください。 .p7b、.p7c、または類似のファイル拡張子で終了するファイルは使用しないでください。

5. 新しい CA 署名証明書と任意の中間証明書を /etc/pki/tls/certs ディレクトリに配置します。

Note

EC2 インスタンスに新しい証明書をアップロードする方法は複数ありますが、最も簡単でわかりやすい方法は、テキストエディタ (vi、nano、またはメモ帳など) をローカルコンピュータとインスタンスの両方で開いて、両者の間でファイルの内容をコピーして貼り付けることです。EC2 インスタンス内でこれらの操作を実行する際には、root [sudo] アクセス許可が必要です。こうすることで、許可やパスに問題があるかどうかをすぐに確認できます。ただし、内容をコピーする際に行を追加したり、内容を変更したりしないでください。

/etc/pki/tls/certs ディレクトリ内から、ファイルの所有権、グループ、アクセス権の設定が制限の厳しい AL2023 のデフォルト (所有者 = root、グループ = root、所有者のみが読み込み/書き込み可) と一致することを確認します。以下の例では、使用するコマンドを示しています。

```
[ec2-user certs]$ sudo chown root:root custom.crt  
[ec2-user certs]$ sudo chmod 600 custom.crt  
[ec2-user certs]$ ls -al custom.crt
```

これらのコマンドによって、次の結果が得られます。

```
-rw----- root root custom.crt
```

中間証明書ファイルのアクセス権は、比較的厳しくありません (所有者 = root、グループ = root、所有者による書き込み可、グループによる読み取り可、その他による読み取り可)。以下の例では、使用するコマンドを示しています。

```
[ec2-user certs]$ sudo chown root:root intermediate.crt
[ec2-user certs]$ sudo chmod 644 intermediate.crt
[ec2-user certs]$ ls -al intermediate.crt
```

これらのコマンドによって、次の結果が得られます。

```
-rw-r--r-- root root intermediate.crt
```

6. CSR の作成に使用したプライベートキーを /etc/pki/tls/private/ ディレクトリに配置します。

 Note

EC2 インスタンスにカスタムキーをアップロードする方法は複数ありますが、最も簡単でわかりやすい方法は、テキストエディタ (vi、nano、メモ帳など) をローカルコンピュータとインスタンスの両方で開いて、両者の間でファイルの内容をコピーして貼り付けることです。EC2 インスタンス内でこれらの操作を実行する際には、root [sudo] アクセス許可が必要です。こうすることで、許可やパスに問題があるかどうかをすぐに確認できます。ただし、内容をコピーする際に行を追加したり、内容を変更したりしないでください。

/etc/pki/tls/private ディレクトリ内から以下のコマンドを使用して、ファイルの所有権、グループ、アクセス権の設定が制限の厳しい AL2023 のデフォルト (所有者 = root、グループ = root、所有者のみが読み込み/書き込み可) と一致することを検証します。

```
[ec2-user private]$ sudo chown root:root custom.key
[ec2-user private]$ sudo chmod 600 custom.key
[ec2-user private]$ ls -al custom.key
```

これらのコマンドによって、次の結果が得られます。

```
-rw----- root root custom.key
```


7. 新しい証明書とキーファイルに合わせるには、`/etc/httpd/conf.d/ssl.conf` を編集します。

 Note

SSLEngine on が VirtualHost ブロックに設定されていることを確認します。

- a. CA 署名のホスト証明書のパスとファイル名を Apache の SSLCertificateFile ディレクティブで指定します。

```
SSLCertificateFile /etc/pki/tls/certs/custom.crt
```

- b. 中間証明書ファイル (この例では `intermediate.crt`) を受け取ったら、Apache の SSLCACertificateFile ディレクティブを使用して、次のファイルのパスとファイル名を指定します。

```
SSLCACertificateFile /etc/pki/tls/certs/intermediate.crt
```

 Note

一部の CA では、ホスト証明書と中間証明書を組み合わせて 1 つのファイルを作成するため、この SSLCACertificateFile ディレクティブは必要ありません。CA が提供している手順を参照してください。

- c. プライベートキー (この例では `custom.key`) のパスとファイル名を Apache の SSLCertificateKeyFile ディレクティブで指定します。

```
SSLCertificateKeyFile /etc/pki/tls/private/custom.key
```

8. `/etc/httpd/conf.d/ssl.conf` を保存して、Apache を再起動します。

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

9. サーバーをテストするには、ブラウザの URL バーにドメイン名を入力し、プレフィックス `https://` を指定します。ブラウザによって、エラーが生成されることなく、HTTPS 経由でテストページがロードされます。

ステップ 3: セキュリティ設定をテストして強化する

TLS が運用可能になりパブリックに公開されたら、実際の安全性をテストする必要があります。セキュリティセットアップの詳細な分析を無料で行うことのできる [Qualys SSL Labs](#) などのオンラインサービスを使用すると簡単です。その結果に基づき、受け入れるプロトコル、優先する暗号化方式、除外する暗号化方式を制御することによって、デフォルトのセキュリティ設定を強化するかどうかを決定できます。詳細については、「[Qualys のスコアの計算方法](#)」を参照してください。

Important

サーバーのセキュリティを確保するには、実際のテストが非常に重要です。小さな設定エラーによって、深刻なセキュリティ侵害やデータの損失が生じる可能性があります。調査や新たな脅威に応じて、推奨されるセキュリティ管理方法は常に変化するため、適切なサーバー管理を行うには、定期的なセキュリティ監査が不可欠です。

[Qualys SSL Labs](#) のサイトで、サーバーの完全修飾ドメイン名を **www.example.com** という形式で入力します。約 2 分後に、サイトに関するグレード (A から F) と、結果の詳細な内訳が届きます。以下の表は、AL2023 でのデフォルトの Apache 設定およびデフォルトの Certbot 証明書と同じ設定を使用しているドメインのレポートをまとめたものです。

総合評価	B
証明書	100%
プロトコルサポート	95%
キー交換	70%
暗号強度	90%

概要は設定がほとんど正常であることを示していますが、詳細レポートでは、いくつかの潜在的な問題が指摘されています。重大度の高い順に以下に示します。

RC4 暗号は、特定の古いブラウザでの使用がサポートされています。暗号は、暗号化アルゴリズムの計算の中核です。TLS データストリームの暗号化に使用される高速の暗号化方式である RC4 は、いくつかの [重大な脆弱性](#) を持つことで知られています。従来のブラウザをサポートするもっともな理由がない限り、この暗号化方式を無効にする必要があります。

x旧バージョンの TLS がサポートされています。設定では TLS 1.0 (すでに廃止されています) と TLS 1.1 (廃止予定) がサポートされています。2018 年以降は、TLS 1.2 のみ推奨されています。

前方秘匿性は完全にサポートされていません。[前方秘匿性](#)は、プライベートキーから派生した一時 (エフェメラル) セッションキーを使用して暗号化を行う、アルゴリズムの機能です。これは、攻撃者がウェブサーバーの長期的なプライベートキーを所有していても、HTTPS データを復号できないことを意味します。

TLS 設定を修正し、将来への対応性を確保するには

1. 設定ファイル `/etc/httpd/conf.d/ssl.conf` を開き、行頭に `#` を付けて以下の行をコメントアウトしてください。

```
#SSLProtocol all -SSLv3
```

2. 次のディレクティブを追加します。

```
#SSLProtocol all -SSLv3
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
```

このディレクティブにより、SSL バージョン 2、3、および TLS バージョン 1.0、1.1 が明示的に無効化されます。これで、サーバーでは、TLS 1.2 以外を使用した、クライアントとの暗号化された接続の受け入れが拒否されます。ディレクティブに含める指定が多くなるほど、サーバーの動作に対する設定内容が明確に伝わります。

Note

このようにして、TLS バージョン 1.0 および 1.1 を無効にすると、ごく一部の古くなったウェブブラウザによるサイトへのアクセスがブロックされるようになります。

許可された暗号のリストを変更するには

1. 設定ファイル `/etc/httpd/conf.d/ssl.conf` で、**SSLCipherSuite** ディレクティブを含むセクションを探し、行頭に `#` を付けて既存の行をコメントアウトします。

```
#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

2. 明示的な暗号スイートと、前方秘匿性を優先し、安全でない暗号を禁止する暗号順序を指定します。ここで使用される SSLCipherSuite ディレクティブは、[Mozilla SSL Configuration Generator](#)の出力に基づいています。これは、お客様のサーバーで実行されている特定のソフトウェアに合わせて TLS 設定を調整します。まず、以下のコマンドの出力を使用して、Apache と OpenSSL のバージョンを確認します。

```
[ec2-user ~]$ yum list installed | grep httpd
```

```
[ec2-user ~]$ yum list installed | grep openssl
```

例えば、返された情報が Apache 2.4.34 および OpenSSL 1.0.2 である場合、これをジェネレーターに入力します。"最新" 互換性モデルを選択すると、SSLCipherSuite ディレクティブが作成されます。このディレクティブは、積極的にセキュリティを適用しますが、ほとんどのブラウザで使用できます。ソフトウェアで最新互換性モデルがサポートされていない場合は、ソフトウェアを更新するか、"中間" の構成を選択します。

```
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
```

選択された暗号化方式の名前には、ECDHE が含まれています (Elliptic Curve Diffie-Hellman Ephemeral の略語です)。ephemeral は前方秘匿性を示します。また、これらの暗号化方式では、RC4 はサポートされていません。

デフォルトや、内容が見えない簡単なディレクティブに依存するのではなく、暗号化方式の明示的なリストを使用することをお勧めします。

生成されたディレクティブを `/etc/httpd/conf.d/ssl.conf` にコピーします。

Note

ここでは読みやすくするために数行に分けて示していますが、このディレクティブは、`/etc/httpd/conf.d/ssl.conf` にコピーする際に、暗号化方式名の間をコロンのみ (スペースなし) で区切り、1 行に指定する必要があります。

3. 最後に、次の行について、行頭の `#` を削除してコメント解除します。

```
#SSLHonorCipherOrder on
```

このディレクティブは、(この場合) 前方秘匿性をサポートするものも含めて、ランクの高い暗号化方式を優先するようサーバーに強制します。このディレクティブが有効になると、サーバーは、セキュリティの弱い暗号化方式に戻る前に、セキュリティが強力な接続を確立しようとしています。

これらの手順がいずれも完了したら、変更内容を `/etc/httpd/conf.d/ssl.conf` に保存し、Apache を再起動します。

[Qualys SSL Labs](#) でドメインをもう一度テストすると、RC4 脆弱性やその他の警告は解決し、次のようなサマリレポートが出力されます。

総合評価	A
証明書	100%
プロトコルサポート	100%
キー交換	90%
暗号強度	90%

OpenSSL の更新ごとに、新しい暗号化方式が導入され古い暗号化方式のサポートが削除されます。EC2 の AL2023 インスタンスを最新の状態に維持し、セキュリティに関する [OpenSSL](#) からの告知に注意して、技術分野の報道でセキュリティ面の新しい脆弱性に関する報告に注意する必要があります。

トラブルシューティング

- パスワードを指定しないと Apache ウェブサーバーが起動しません

これは、パスワードで保護された暗号化プライベート サーバー キーをインストールした場合は正常な動作です。

暗号化とパスワードの要件をキーから削除できます。デフォルトディレクトリに `custom.key` という暗号化プライベート RSA キーがあり、そのパスワードが **abcde12345** であるとする

と、EC2 インスタンスで次のコマンドを実行し、このキーの非暗号化バージョンを生成してください。

```
[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
```

パスワードが求められずに Apache が起動するようになります。

- `sudo dnf install -y mod_ssl` を実行するとエラーが発生します。

SSL に必要なパッケージをインストールすると、次のようなエラーが表示されることがあります。

```
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
```

これは通常、EC2 インスタンスで AL2023 が実行されていないことを意味します。このチュートリアルでは、公式の AL2023 AMI から新しく作成されたインスタンスのみをサポートしています。

チュートリアル: AL2023 で WordPress ブログをホストする

次の手順では、ユーザーの AL2023 インスタンスで、WordPress ブログのインストール、構成を実行し、安全性を確立します。このチュートリアルは、WordPress ブログをホストするウェブサーバーを完全に制御する (これは従来のホスティングサービスでは一般的なことはありません) という点で、Amazon EC2 を使用するための優れた手引きになります。

サーバーに対するソフトウェアパッケージの更新とセキュリティパッチの維持は、お客様の責任となります。ウェブサーバー設定と直接やり取りする必要のない、より自動化された WordPress インストールの場合、この CloudFormation サービスには WordPress テンプレートが用意されており、すぐに開始することもできます。詳細については、AWS CloudFormation ユーザーガイドの「[開始方法](#)」を参照してください。データベースが疎結合化された高可用性のソリューションが必要な場合

は、AWS Elastic Beanstalk デベロッパーガイドの「[高可用性の WordPress ウェブサイトをデプロイする](#)」を参照してください。

Important

これらの手順は AL2023 で使用するためのものです。その他のディストリビューションの情報については各ドキュメントを参照してください。このチュートリアル多くの手順は、Ubuntu インスタンスには使用できません。Ubuntu インスタンスでの WordPress のインストールについては、Ubuntu のドキュメントの「[WordPress](#)」を参照してください。[CodeDeploy](#) を使用して、Amazon Linux、macOS、または Unix システムでこのタスクを実行することもできます。

トピック

- [前提条件](#)
- [WordPress のインストール](#)
- [次のステップ](#)
- [ヘルプ! パブリック DNS 名が変更されたため、ブログが壊れました](#)

前提条件

Elastic IP アドレス (EIP) は、WordPress ブログのホストに使用しているインスタンスに関連付けることを強くお勧めします。これにより、インスタンスのパブリック DNS アドレスが変更されて、インストールが破損することを防止できます。ドメイン名を所有していてそのドメインをブログに使用する場合、EIP アドレスをポイントするようにドメイン名の DNS レコードを更新できます (これを行うには、ドメイン名レジストラに問い合わせてください)。実行中のインスタンスに関連付けられた EIP アドレスを無料で 1 つ取得できます。詳細については、「Amazon EC2 ユーザーガイド」の「[Elastic IP アドレス](#)」を参照してください。[チュートリアル: AL2023 に LAMP サーバーをインストールする](#) チュートリアルでは、セキュリティグループで HTTP および HTTPS トラフィックを許可するように設定する手順や、ウェブサーバー用にファイル許可が正しく設定されていることを確認する手順も示します。セキュリティグループへのルールの追加の詳細については、「[セキュリティグループへのルールの追加](#)」を参照してください。

ブログのドメイン名がまだない場合は、Route 53 にドメイン名を登録し、そのドメイン名にインスタンスの EIP アドレスを関連付けることができます。詳細については、Amazon Route 53 デベロッパーガイドの「[Amazon Route 53 を使用したドメイン名の登録](#)」を参照してください。

WordPress のインストール

インスタンスに接続して、WordPress インストールパッケージをダウンロードします。インスタンスへの接続の詳細については[AL2023 インスタンスへの接続](#)を参照してください。

1. 次のコマンドを使用してこれらのパッケージをダウンロードしてインストールします。

```
dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json  
php php-devel -y
```

2. 出力に同様の警告文が表示されることがあります (バージョンは表示されるタイミングによって異なる場合があります)。

```
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
```

ベストプラクティスとして、OS を可能な限り最新の状態にしておくことをお勧めします。ただし、環境内で競合が発生しないように各バージョンをイテレーションすることをお勧めします。ステップ 1 に記載されている前述のパッケージのインストールが失敗した場合は、リストされているいずれかの新しいリリースに更新して、再試行する必要がある場合があります。

3. wget コマンドを使って、最新の WordPress インストールパッケージをダウンロードします。次のコマンドを実行すると、最新リリースが必ずダウンロードされます。

```
[ec2-user ~]$ wget https://wordpress.org/latest.tar.gz
```

4. インストールパッケージを解凍します。インストールフォルダは、wordpress という名前のフォルダに解凍されます。


```
[ec2-user ~]$ tar -xzf latest.tar.gz
```

WordPress インストール用にデータベースユーザーとデータベースを作成するには

WordPress インストールは、ブログの投稿、ユーザーコメントなどの情報をデータベースに格納する必要があります。この手順を実行すると、ブログのデータベースを作成するのに役立ち、このデータベースに対して情報の読み取りや保存を許可されたユーザーにも有用です。

1. データベースおよびウェブサーバーを起動します。

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

2. データベースサーバーに root ユーザーとしてログインします。メッセージが表示されたら、データベース root パスワードを入力します。これは通常の root システムパスワードと異なることもあれば、データベースサーバーのセキュリティ確保を実行していない場合は、空のときもあります。

データベースサーバーのセキュリティを確保していない場合、セキュリティ確保を行うことは重要です。詳細については、[ステップ 3: データベースサーバーをセキュリティで保護する \(AL2023\)](#) を参照してください。

```
[ec2-user ~]$ mysql -u root -p
```

3. MySQL データベースのユーザーとパスワードを作成します。WordPress インストールは、これらの値を使って、MySQL データベースと通信を行います。一意のユーザー名とパスワードを入力して、次のコマンドを入力します。

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

ユーザー用に強力なパスワードを作成してください。パスワードに一重引用符 (') を使用しないでください。この文字は前述のコマンドを中断させるためです。既存のパスワードを再利用しないでください。また、このパスワードは必ず安全な場所に保管してください。

4. データベースを作成します。wordpress-db など、データベースにはわかりやすい名前を使用します。

Note

次のコマンドのデータベース名を囲む区切り記号は、「バックティック」と呼ばれています。バックティック(`)キーは通常、標準キーボードの Tab キーの上に配置されています。バックティックは必ずしも必要ではありませんが、データベース名では使用できない文字(ハイフンなど)の代わりに使用できます。

```
CREATE DATABASE `wordpress-db`;
```

5. データベースに対して、以前作成した WordPress ユーザーに対する完全な権限を付与します。

```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

6. すべての変更を有効にするため、データベース権限をフラッシュします。

```
FLUSH PRIVILEGES;
```

7. mysql クライアントを終了します。

```
exit
```

wp-config.php ファイルの作成と編集を行うには

WordPress インストールフォルダには、wp-config-sample.php という名前の構成ファイル例が格納されています。この手順では、このファイルをコピーして、特定の構成に合うように編集します。

1. wp-config-sample.php ファイルを wp-config.php という名前でコピーします。この操作を実行すると、新しい構成ファイルが作成され、元のファイルがバックアップとしてそのまま保持されます。

```
[ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
```

2. お好みのテキストエディタ(wp-config.php、nano など)を使って vim ファイルを編集し、インストール用の値を入力します。お好みのテキストエディタがない場合、nano が初心者に適しています。

```
[ec2-user ~]$ nano wordpress/wp-config.php
```

- a. DB_NAME を定義する行を探して、database_name_here を [Step 4 の WordPress インストール用にデータベースユーザーとデータベースを作成するには](#) で作成したデータベース名に変更します。

```
define('DB_NAME', 'wordpress-db');
```

- b. DB_USER を定義する行を探して、username_here を [Step 3 の WordPress インストール用にデータベースユーザーとデータベースを作成するには](#) で作成したデータベースユーザーに変更します。

```
define('DB_USER', 'wordpress-user');
```

- c. DB_PASSWORD を定義する行を探して、password_here を [Step 3 の WordPress インストール用にデータベースユーザーとデータベースを作成するには](#) で作成した強力なパスワードに変更します。

```
define('DB_PASSWORD', 'your_strong_password');
```

- d. Authentication Unique Keys and Salts というセクションを見つけます。これらの KEY と SALT の値は、WordPress ユーザーがローカルマシンに保存したブラウザクッキーに対する暗号化レイヤーを提供します。基本的に、ここで長くてランダムな値を指定すると、サイトのセキュリティが向上します。<https://api.wordpress.org/secret-key/1.1/salt/> にアクセスして、ランダムに生成されるキーセット値を取得し、wp-config.php ファイルにコピーして貼り付けることができます。PuTTY 端末にテキストを貼り付けるには、テキストを貼り付ける場所にカーソルを置き、PuTTY 端末内でマウスを右クリックします。

セキュリティキーの詳細については、「<https://wordpress.org/support/article/editing-wp-config-php/#security-keys>」にアクセスしてください。

 Note

次の値はサンプル専用です。これらの値を実際のインストールには使わないでください。

```
define('AUTH_KEY',          ' #U$$+[RXN8:b^-L 0(WU_+ c+WFkI~c]o]-bHw+)/
Aj[wTwSiZ<Qb[mghEXcRh-');
define('SECURE_AUTH_KEY',   'Zsz._P=l/|y.Lq)XjlkW51y5NJ76E6EJ.AV0pCKZZB,*~*r ?
60P$eJT@;+(ndLg');
define('LOGGED_IN_KEY',     'ju}qwre3V*+8f_z0Wf?{LLGsQ]Ye@2Jh^,8x>)Y |;(^[Iw]Pi
+LG#A4R?7N`YB3');
define('NONCE_KEY',         'P(g62HeZxEes|LnI^i=H,[XwK9I&[2s|:~0N}VJM%?;v2v]v+;
+^9eXUahg@::Cj');
define('AUTH_SALT',         'C$DpB4Hj[JK:~{q1`sRVa:{:7yShy(9A@5wg+`JJVb1fk%-
Bx*M4(qc[Qg%JT!h');
define('SECURE_AUTH_SALT',  'd!uRu#}+q#{f$Z?Z9uFPG.${+S{n~1M&%@~gL>U>NV<zpD-@2-
Es7Q10-bp28EKv');
define('LOGGED_IN_SALT',    ';j{00P*owZf)kVD+FVLn-~ >.|Y%Ug4#I^*LVd9QeZ^&XmK|
e(76miC+&W&+^0P/');
define('NONCE_SALT',        '-97r*V/cgxLmp?Zy4zUU4r99QQ_rGs2LTd%P;|
_e1tS)8_B/,.6[=UK<J_y9?JWG');
```

- e. ファイルを保存し、テキストエディタを終了します。

WordPress ファイルを Apache ドキュメントルートの下にインストールするには

- インストールフォルダの解凍、MySQL データベースとユーザーの作成、WordPress 構成ファイルのカスタマイズが終了したため、インストールファイルをウェブサーバーのドキュメントルートにコピーし、インストールスクリプトを実行して、インストールを終了する準備ができました。これらのファイルの場所は、ウェブサーバーの実際のルートで WordPress ブログを使用できるようにするかどうか (*my.public.dns.amazonaws.com* など)、またはルートの下サブディレクトリやフォルダに格納するか (*my.public.dns.amazonaws.com/blog* など) によって異なります。
- WordPress をドキュメントルートで実行する場合は、WordPress のインストールディレクトリのコンテンツを次のようにコピーします (ただし、ディレクトリ自体はコピーしません)。

```
[ec2-user ~]$ cp -r wordpress/* /var/www/html/
```

- WordPress をドキュメントルートの下別のディレクトリで実行する場合、まず、そのディレクトリを作成してから、そこにファイルをコピーします。この例では、WordPress はディレクトリ *blog* から実行されます。

```
[ec2-user ~]$ mkdir /var/www/html/blog
[ec2-user ~]$ cp -r wordpress/* /var/www/html/blog/
```

⚠ Important

セキュリティ上の理由から、次の手順にすぐに進まない場合は、Apache ウェブサーバー (httpd) を直ちに停止してください。インストールを Apache ドキュメントルートの下に移動すると、WordPress インストールスクリプトは保護されなくなり、Apache ウェブサーバーが実行している場合、攻撃者はブログへのアクセス権を取得する可能性があります。Apache ウェブサーバーを停止するには、`sudo service httpd stop` コマンドを入力します。次の手順に移動する場合、Apache ウェブサーバーを停止する必要はありません。

WordPress がパーマリンクを使用できるようにするには

WordPress のパーマリンクが正しく機能するには Apache の `.htaccess` ファイルを使用する必要がありますが、Amazon Linux ではデフォルトで有効になっていません。Apache ドキュメントルートですべての上書きできるようにするには、次の手順を使用します。

1. お好みのテキストエディタ (`httpd.conf` や `nano` など) で、`vim` ファイルを開きます。お好みのテキストエディタがない場合、`nano` が初心者に適しています。

```
[ec2-user ~]$ sudo vim /etc/httpd/conf/httpd.conf
```

2. `<Directory "/var/www/html">` で始まるセクションを見つけます。

```
<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. 上のセクションの `AllowOverride None` 行を `AllowOverride All` に変更します。

 Note

このファイルには複数の `AllowOverride` 行があります。必ず `<Directory "/var/www/html">` セクションの行を変更してください。

```
AllowOverride All
```

4. ファイルを保存し、テキストエディタを終了します。

PHP グラフィック描画ライブラリを AL2023 にインストールするには

PHP 用の GD ライブラリを使用すると、イメージを変更することができます。ブログのヘッダーイメージをトリミングする必要がある場合は、このライブラリをインストールします。インストールするバージョンの phpMyAdmin は、このライブラリの特定の最小バージョン (バージョン 8.1 など) を必要とする場合があります。

PHP グラフィック描画ライブラリを AL2023 にインストールするには、次のコマンドを使用します。例えば、LAMP スタックをインストールする一環としてソースから php8.1 をインストールした場合、このコマンドは PHP グラフィック描画ライブラリのバージョン 8.1 をインストールします。

```
[ec2-user ~]$ sudo dnf install php-gd
```

インストールしたバージョンを検証するには、次のコマンドを使用します。

```
[ec2-user ~]$ sudo dnf list installed | grep php-gd
```

出力例を次に示します。

```
php-gd.x86_64                8.1.30-1.amzn2                @amazonlinux
```

PHP グラフィック描画ライブラリを Amazon Linux AMI にインストールするには

PHP 用の GD ライブラリを使用すると、イメージを変更することができます。ブログのヘッダーイメージをトリミングする必要がある場合は、このライブラリをインストールします。インストールするバージョンの phpMyAdmin は、このライブラリの特定の最小バージョン (バージョン 8.1 など) を必要とする場合があります。

使用可能なバージョンを確認するには、次のコマンドを使用します。

```
[ec2-user ~]$ dnf list | grep php
```

PHP グラフィック描画ライブラリ (バージョン 8.1) の出力例を次に示します。

```
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
```

PHP グラフィック描画ライブラリの特定のバージョン (バージョン php8.1 など) を Amazon Linux AMI にインストールするには、次のコマンドを使用します。

```
[ec2-user ~]$ sudo dnf install -y php8.1-gd
```

Apache ウェブサーバーのファイル許可を修正するには

WordPress で使用できる機能の中には、Apache ドキュメントルートへの書き込み権限が必要なものがあります (管理画面を使った、メディアのアップロードなど)。まだ設定していない場合は、次のグループのメンバーシップおよびアクセス許可を適用します (プロセスの詳細は「[LAMP ウェブサーバーチュートリアル](#)」を参照)。

1. /var/www とそのコンテンツのファイル所有権を apache ユーザーに付与します。

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

2. /var/www とそのコンテンツのグループ所有権を apache グループに付与します。

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

3. /var/www およびそのサブディレクトリのディレクトリ許可を変更してグループの書き込み許可を設定し、将来のサブディレクトリにグループ ID を設定します。

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

4. /var/www およびそのサブディレクトリのファイル許可を繰り返し変更します。

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

Note

WordPress を FTP サーバーとして使用する場合も、これよりも制限の少ないグループ設定が必要になります。これを実行するには、「[WordPress で推奨されている手順とセキュリティ設定](#)」を参照してください。

5. Apache ウェブサーバーを再起動して、新しいグループと許可を有効にします。

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


AL2023 で WordPress インストールスクリプトを実行するには

WordPress をインストールする準備ができました。使用するコマンドは、オペレーティングシステムによって異なります。この手順のコマンドは、AL2023 で使用するためのものです。AL2023 AMI で、この後の手順を使用します。

1. `systemctl` コマンドを使って、`httpd` サービスとデータベースサービスがシステムブート時に起動することを確認します。

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

2. データベースサーバーが実行中であることを確認します。

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

データベースサービスが実行されていない場合は、起動します。

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

3. Apache ウェブサーバー (`httpd`) が実行中であることを確認します。

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

`httpd` サービスが実行されていない場合は、起動します。

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

4. ウェブブラウザで WordPress ブログの URL を入力します (インスタンスのパブリック DNS アドレス、または `blog` フォルダに続くアドレス)。WordPress インストールスクリプトが表示されます。WordPress のインストールに必要な情報を入力します。[WordPress のインストール] を選択して、インストールを完了します。詳細については、WordPress のウェブサイトの [Step 5: Run the Install Script](#) を参照してください。

AL2023 AMI で WordPress インストールスクリプトを実行するには

1. `chkconfig` コマンドを使って、`httpd` サービスとデータベースサービスがシステムブート時に起動することを確認します。

```
[ec2-user ~]$ sudo chkconfig httpd on && sudo chkconfig mariadb on
```

2. データベースサーバーが実行中であることを確認します。

```
[ec2-user ~]$ sudo service mariadb status
```

データベースサービスが実行されていない場合は、起動します。

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

3. Apache ウェブサーバー (httpd) が実行中であることを確認します。

```
[ec2-user ~]$ sudo service httpd status
```

httpd サービスが実行されていない場合は、起動します。

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

4. ウェブブラウザで WordPress ブログの URL を入力します (インスタンスのパブリック DNS アドレス、または blog フォルダに続くアドレス)。WordPress インストールスクリプトが表示されます。WordPress のインストールに必要な情報を入力します。[WordPress のインストール] を選択して、インストールを完了します。詳細については、WordPress のウェブサイトの [Step 5: Run the Install Script](#) を参照してください。

次のステップ

WordPress ブログをテストしたら、設定の更新を検討します。

カスタムドメイン名を使用する

EC2 インスタンスの EIP アドレスに関連付けられたドメイン名がある場合、EC2 パブリック DNS アドレスの代わりにその名前を使用するようにブログを設定できます。詳細については、WordPress ウェブサイトの「[サイトの URL の変更](#)」を参照してください。

ブログを設定する

読者にパーソナライズされた体験を提供するため、さまざまな[テーマ](#)や[プラグイン](#)を使用するようにブログを設定できます。ただし、インストールプロセスで問題が発生してブログ全体が失われること

があります。インストール中に問題が発生した場合もブログを復元できるように、テーマやプラグインをインストールする前にインスタンスのバックアップ Amazon マシンイメージ (AMI) を作成しておくことを強くお勧めします。詳細については、「Amazon EC2 ユーザーガイド」の「[独自の AMI の作成](#)」を参照してください。

容量を増やす

WordPress ブログが人気になり処理能力やストレージを増やす必要がある場合は、次のステップを検討してください。

- インスタンスストレージ領域を拡張する。詳細については、「[Amazon EBS Elastic Volumes](#)」を参照してください。
- MySQL データベースを [Amazon RDS](#) に移動して、サービスが持つ容易にスケールする機能を活用する。

インターネットトラフィックのネットワークパフォーマンスを向上させる

ブログにより世界中のユーザーからのトラフィックが増加すると予想される場合は、[AWS Global Accelerator](#) をご検討ください。Global Accelerator を使用すると、ユーザーのクライアントデバイスと AWS で実行中の WordPress アプリケーションとの間で、インターネットトラフィックのパフォーマンスを向上でき、低レイテンシーを実現できます。Global Accelerator は、[AWS グローバルネットワーク](#) を使用して、クライアントに最も近い AWS リージョンの正常なアプリケーションエンドポイントにトラフィックを誘導します。

WordPress の詳細

次のリンクには、WordPress に関する詳細情報が含まれています。

- WordPress の詳細については、「[Codex](#)」にある WordPress Codex ヘルプ文書を参照してください。
- インストールのトラブルシューティングの詳細については、「[インストールに関する一般的な問題](#)」を参照してください。
- WordPress ブログのセキュリティ向上の詳細については、「[WordPress の強化](#)」を参照してください。
- WordPress ブログを最新状態に維持する方法についての詳細は、「[WordPress の更新](#)」を参照してください。

ヘルプ! パブリック DNS 名が変更されたため、ブログが壊れました

WordPress のインストールは、EC2 インスタンスのパブリック DNS アドレスを使用して自動的に設定されます。インスタンスを停止および再開した場合、パブリック DNS アドレスが変更され (Elastic IP アドレスに関連付けられている場合を除く)、ブログが存在しなくなった (または別の EC2 インスタンスに割り当てられた) アドレスにあるリソースを参照することになるため、ブログは機能しなくなります。問題および考えられるいくつかの解決策の詳細については、<https://wordpress.org/support/article/changing-the-site-url/> で説明されています。

WordPress のインストール時にこの状況が発生した場合、WordPress の wp-cli コマンドラインインターフェイスを使用する以下の手順でブログを復元できる可能性があります。

wp-cli を使用して WordPress のサイト URL を変更するには

1. SSH を使って EC2 インスタンスに接続します。
2. インスタンスの古いサイト URL と新しいサイト URL を書き留めます。古いサイト URL は、WordPress をインストールした時点での EC2 インスタンスのパブリック DNS 名と考えられます。新しいサイト URL は、EC2 インスタンスの現在のパブリック DNS 名です。古いサイト URL が不明な場合、次のコマンドで curl を使用して調べることができます。

```
[ec2-user ~]$ curl localhost | grep wp-content
```

古いパブリック DNS 名への参照が出力に表示されます。次に例を示します (古いサイト URL は赤色になっています)。

```
<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. 次のコマンドを使って wp-cli をダウンロードします。

```
[ec2-user ~]$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
```

4. 次のコマンドを使って、WordPress インストールの古いサイト URL を検索し、置き換えます。EC2 インスタンスの古いサイト URL と新しいサイト URL、および WordPress のインストールパス (通常は /var/www/html または /var/www/html/blog) を置き換えます。

```
[ec2-user ~]$ php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/path/to/wordpress/installation --skip-columns=guid
```

5. ウェブブラウザで、WordPress ブログの新しいサイト URL を入力し、サイトが再び正しく動作していることを確認します。正しく動作していない場合は、[「サイトの URL の変更」](#)と [「一般的なインストールの問題」](#)を参照してください。

チュートリアル: AL2023 での Redis 6 から Valkey への移行

次のドキュメントでは、AL2023 での Redis 6 から Valkey への移行の主な側面について説明します。

Redis 6 のサポートタイムライン

Redis 6 は 2027 年 1 月 31 日にサポート終了 (EOL) に達しました。この日付が過ぎると、Redis 6 は Redis プロジェクトから更新プログラムやセキュリティパッチを受信しなくなります。継続的なサポートとセキュリティ更新を確保するために、ユーザーは 2027 年 1 月より前に Valkey に移行することを強くお勧めします。

Redis バージョンのサポートタイムラインの詳細については、[Redis のサポート終了スケジュール](#)に関するドキュメントを参照してください。

Valkey の概要

Valkey は Redis 7 のオープンソースフォークであり、Linux Foundation によって管理されています。Redis オープンソースソフトウェア (OSS) バージョン 2.x ~ 7.2.x と完全に互換性があります。Valkey は、使い慣れた Redis API と機能を維持する一方で、以下のいくつかの機能強化を提供します。

- マルチスレッドによるパフォーマンスの向上。
- メモリ効率の向上 (特にクラスターモードで)。
- デュアルチャネルレプリケーションによるデータ整合性の向上。

移行の計画とタイムライン

Redis 6 がサポート終了 (EOL) に達した 2027 年 1 月 31 日以前に Redis 6 から Valkey に移行することを強くお勧めします。この移行には手動介入が必要であり、自動ではありません。

Amazon Linux では、Redis に依存するアプリケーションの機能、サポート、セキュリティ更新を継続的に確保するために、この移行を推奨しています。

移行のオプションとステップ

デプロイ要件と運用ニーズに基づいて、Valkey への 3 つの移行パスを提案します。

オプション 1: 新しいインスタンスのインストール

新しいデプロイの場合、またはデータ移行が必要ない場合:

1. Valkey をインストールします。

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

2. Valkey を起動します。

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

3. (オプション) 起動時に Valkey を有効にします。

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

4. インストールを確認します。

```
[ec2-user ~]$ valkey-cli info server  
[ec2-user ~]$ valkey-cli ping
```

オプション 2: インプレース置換

データ永続性が不要な既存のインスタンスの場合:

1. Redis 6 を停止します。

```
[ec2-user ~]$ sudo systemctl stop redis6
```

2. Valkey をインストールします。

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

3. (オプション) Valkey で Redis 6 設定を使用します。

```
[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. (オプション) Valkey で Redis 6 センチネル設定ファイルを使用します。

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

5. Valkey を起動します。

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

6. (オプション) 起動時に Valkey を有効にします。

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

7. Valkey のインストールを確認します。

```
[ec2-user ~]$ valkey-cli info server
[ec2-user ~]$ valkey-cli ping
```

8. Redis 6 を削除します。

```
[ec2-user ~]$ sudo dnf remove redis6
```

オプション 3: データ移行

このオプションを使用すると、Redis 6 と Valkey の両方を同時に実行できます。

1. Redis 6 を削除せずに Valkey をインストールします。

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

2. (オプション) Valkey で Redis 6 設定を使用します。

```
[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. (オプション) Valkey で Redis 6 センチネル設定ファイルを使用します。

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

4. Valkey 設定を変更します。

Redis 6 との競合を避けるために、/etc/valkey/valkey.conf を編集して「port」ディレクティブを別の値 (6380 など) に設定します。

5. Valkey を起動します。

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

6. (オプション) 起動時に Valkey を有効にします。

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

7. Valkey のインストールを確認します。

```
[ec2-user ~]$ valkey-cli -p port info server
[ec2-user ~]$ valkey-cli -p port ping
```

Note

port を、設定したポート番号に置き換えます。

8. データを移行します。

これで、レプリケーションまたは手動データ転送方法を使用して、Redis 6 から Valkey にデータを移行できます。

9. アプリケーション設定を更新します。

Valkey ポートを使用するようにアプリケーションを段階的に更新します。

10. Redis 6 を削除します。

すべてのデータとアプリケーションを移行したら、Redis 6 を停止して削除できます。


```
[ec2-user ~]$ sudo systemctl stop redis6  
[ec2-user ~]$ sudo dnf remove redis6
```

Note

本番環境システムに変更を実装する前に、テスト環境で移行プロセスを検証することを強くお勧めします。

関連トピック

Valkey の詳細については、以下を参照してください。

- Valkey: <https://valkey.io/>
- Valkey 移行: <https://valkey.io/topics/migration/>

チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする

[GNOME デスクトップ環境](#)は、AL2023 のリリース 2023.7 以降でオプションのグラフィカルユーザーインターフェイスとして使用できます。

次の手順で、AL2023 インスタンスに GNOME デスクトップ環境をインストールできます。このグラフィカルインターフェイスを使用して、コマンドラインインターフェイスだけでなく、使い慣れたデスクトップ環境を使用して Linux システムとやり取りできます。

内容

- [前提条件](#)
- [インストール](#)
- [関連トピック](#)

前提条件

- デスクトップ環境には、少なくとも 2.4 GB のメモリが必要です。したがって、適切なパフォーマンスを確保するために、タイプ t2.medium 以上のインスタンスが推奨されます。メモリ不足のインスタンスタイプの例として、t2.nano、t2.micro、および t2.small があります。この制限は、このサイズの t3 および t4 インスタンスのほか、メモリ要件を満たさないその他のインスタンスタイプにも適用されます。
- このチュートリアルでは、リリース 2023.7 以降を実行する AL2023 を使用してインスタンスを既に起動していることを前提としています。詳細については、「[Amazon EC2 での AL2023](#)」および「[AL2023 の更新](#)」のページを参照してください。

インストール

- GNOME デスクトップ環境および関連するパッケージをインストールします。

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

Note

グラフィカルデスクトップ環境にアクセスするには、Amazon DCV や VNC などの追加のソフトウェアをインストールして設定する必要があります。これらのツールを使用すると、ネットワーク経由でグラフィカルユーザーインターフェイスに接続して操作できます。

関連トピック

グラフィカルデスクトップ環境の詳細については、次のドキュメントを参照してください。

- 「Amazon DCV 管理者ガイド」の「[Amazon DCV とは](#)」
- [チュートリアル: AL2023 で TigerVNC サーバーを設定する](#)

チュートリアル: AL2023 で TigerVNC サーバーを設定する

以下の手順は、AL2023 インスタンスで VNC サーバーを設定するのに役立ちます。VNC を使用すると、安全なネットワーク接続を介してグラフィカルデスクトップ環境にリモートからアクセスして操作できます。

内容

- [前提条件](#)
- [ステップ 1: インストール](#)
- [ステップ 2: 設定](#)
- [ステップ 3: VNC クライアントを使用して接続する](#)
- [\(オプション\) 起動時にサービスを開始する](#)
- [\(オプション\) アイドルロック画面を無効にする](#)
- [関連トピック](#)

前提条件

- このチュートリアルでは、AL2023 インスタンスに GNOME デスクトップ環境をインストール済みであることを前提としています。詳細については、「[チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする](#)」のページを参照してください。
- このチュートリアルでは、SSH ポート転送を使用して VNC サーバーにアクセスします。キーペアの設定の詳細については、「Amazon EC2 ユーザーガイド」の「[SSH を使用して Linux インスタンスに接続する](#)」を参照してください。
- 次の手順は、VNC クライアントのインストールプロセスを説明するものではありません。デスクトップ環境に接続して操作できるようにするには、ローカルマシンに VNC クライアントをインストールする必要があります。

ステップ 1: インストール

1. インスタンスに接続します。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。
2. AL2023 用の TigerVNC サーバーパッケージをインストールします。

-y オプションを指定すると、確認メッセージを表示せずに更新をインストールします。インストール前にパッケージの内容を確認する場合は、このオプションを省略できます。

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

ステップ 2: 設定

1. ユーザーが VNC パスワードを設定していることを確認します。

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

2. ユーザーにディスプレイ番号を割り当てます。

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver.users
```

次の設定を追加します。

```
:1=ec2-user
```

Note

任意のディスプレイ番号をユーザーに割り当てることができます。この例では、ディスプレイ :1 を使用しています。

3. VNC サーバー設定ファイルを編集します。

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver-config-defaults
```

次の設定を追加します。

```
session=gnome
securitytypes=vncauth,tlsvnc
geometry=1920x1080
localhost
alwaysshared
```

 Note

ディスプレイの解像度は、`geometry` パラメータを使用して変更できます。この例では `1920x1080` を使用しています。

4. VNC サーバーを起動します。このプロセスは、インスタンスを再起動するたびに繰り返す必要があります。このサービスを起動するプロセスを自動化する場合は、以下のオプションの項目を参照してください。

```
[ec2-user ~]$ sudo systemctl start vncserver@:1
```

 Important

`vncserver` サービスを起動する場合、`@` の後の部分は、`/etc/tigervnc/vncserver.users` ファイルでユーザーに設定したディスプレイ番号と一致する必要があります。

このステップを実行すると、ローカルマシンから SSH トンネルを作成し、VNC クライアントを使用して接続できます。

ステップ 3: VNC クライアントを使用して接続する

VNC サーバーは、クライアント接続用の TCP ソケットを公開します。セキュリティグループを通じて VNC ポートを直接公開できますが、このチュートリアルでは、より安全なアプローチとして SSH トンネリングを使用し、ローカルマシンと EC2 インスタンス間の接続を暗号化します。トンネルを介して接続すると、前のステップで設定したパスワードを使用して VNC サーバーに認証します。セキュリティグループの詳細については、「Amazon EC2 ユーザーガイド」の「[Amazon EC2 インスタンスのセキュリティグループの変更](#)」を参照してください。

1. ローカルマシンから SSH トンネルを作成します。

```
$ ssh -i <keypair> -L 5901:localhost:5901 ec2-user@<address>
```

 Note

<keypair> を SSH キーへのパスに置き換え、 <address> をインスタンスのパブリック IP または DNS 名に置き換えます。ポートは、 vncserver の起動に使用したディスプレイ番号に応じて変わります。例えば、ディスプレイ :1 はポート 5901 を使用し、ディスプレイ :2 はポート 5902 を使用します。

2. VNC クライアントで、以前に設定した VNC パスワードを使用して localhost:5901 または 127.0.0.1:5901 に接続します。

 Important

VNC の使用中は、SSH トンネルを開いたままにします。SSH トンネルが開いていない場合、VNC クライアントを使用してデスクトップ環境を表示および操作することはできません。

(オプション) 起動時にサービスを開始する

VNC を定期的に使用する予定がある場合は、インスタンスの起動時に VNC サーバーが自動的に開始するように設定できます。これにより、インスタンスを再起動するたびに VNC サーバーを手動で開始する必要がなくなります。この設定により、インスタンスが起動プロセスを完了するとすぐに、グラフィカルデスクトップ環境の準備が完了し、アクセス可能になります。

- 起動時にサービスを開始するように設定します。

```
[ec2-user ~]$ sudo systemctl enable vncserver@:1
```

 Important

vncserver サービスを有効にする場合、 @ の後の部分は、 /etc/tigervnc/vncserver.users ファイルでユーザーに設定したディスプレイ番号と一致する必要があります。さらに、enable の後に --now 引数を渡して、すぐにサービスを開始することもできます。

このステップを実行すると、インスタンスを再起動するたびに `vncserver` を開始する必要がなくなります。

(オプション) アイドルロック画面を無効にする

- ユーザーが長時間非アクティブであるときに画面がロックされないようにするには、アイドル遅延をゼロに設定します。

```
[ec2-user ~]$ gsettings set org.gnome.desktop.session idle-delay 0
```

関連トピック

グラフィカルデスクトップ環境の詳細については、次のドキュメントを参照してください。

- [チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする](#)
- 「Amazon DCV 管理者ガイド」の「[Amazon DCV とは](#)」

AL2023 カーネルでの多世代 LRU (MGLRU) の使用

[多世代 LRU](#) は、Linux カーネルの最新のページ再利用アルゴリズムであり、メモリに負荷がかかっている状態でメモリ管理のパフォーマンスが向上するように設計されています。このアルゴリズムによって、システムがメモリ不足になったときに再利用するメモリページを決定する従来の LRU (Least Recently Used) メカニズムが置き換えられます。

従来の LRU メカニズムでは、2 リストモデル (アクティブと非アクティブ) を使用してページの使用状況を追跡しているため、大規模な作業セットを伴う最新のワークロードでは効率が低下する場合があります。MGLRU では、これを複数の「世代」のページに置き換え、カーネルがよりきめ細かな経過時間情報に基づいてより賢明な決定を下せるようにします。

MGLRU には次のような利点があります。

- より適切な再利用の決定: コールド (未使用) ページをより正確に識別する。
- レイテンシーの短縮とスループットの向上: 特に、アドレススペースが大きいワークロードや多数の同時プロセスがあるワークロードの場合。
- キャッシュ保持の改善: 最近使用されたページを早期に削除されにくくする。

- スケーラブルでロック効率の高い設計: CPU が多いマシンでパフォーマンスが向上する。

設定と調整

カーネル設定 CONFIG_LRU_GEN は AL2023 カーネルで有効になっています。これは MGLRU でコンパイルされますが、デフォルトでは有効になりません。

MGLRU は、`/sys/kernel/mm/lru_gen/enabled` ファイルを使用して有効化および調整できます。値はビットマスクです。一部のコンポーネントに望ましくない副作用がない限り、すべてのコンポーネントを有効にすることをお勧めします。

Bit	コンポーネント
0	多世代 LRU のメインスイッチ。
1	(x86 など) MMU にこれを設定すると、リーフページテーブルエントリのアクセス済みビットが大規模なバッチでクリアされます。この動作は、理論的にはロック競合を悪化させる可能性があります (<code>mmap_lock</code>)。これを無効にした場合、ホットページを連続してマッピングするワークロードにおいて多世代 LRU のパフォーマンスがわずかに低下し、ホットページのアクセス済みビットが、より少ない数のより大規模なバッチでクリアされる可能性があります。
2	(x86 など) MMU にこれを設定すると、リーフ以外のページテーブルエントリのアクセス済みビットもクリアされます。この動作は、Intel および AMD 以外の種類の x86 では検証されていません。これを無効にした場合、多世代 LRU のパフォーマンスはごくわずかですが低下します。
[yYnN]	上記のすべてのコンポーネントを有効または無効にします。

MGLRU を有効にする方法の例を以下に示します。

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

以下により、すべてのコンポーネントが有効になります。

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

チュートリアル: AL2023 に LibreOffice をインストールする

LibreOffice は、無料のオープンソースのオフィス生産性スイートで、単語処理、スプレッドシート、プレゼンテーション、図、データベース、数式編集用のアプリケーションが含まれています。LibreOffice を Amazon Linux 2023 (AL2023) インスタンスにインストールして、ユーザーにフル機能のデスクトップオフィススイートを提供できます。

LibreOffice は、The Document Foundation によって AL2023 で公式にテストおよびサポートされています。詳細については、[LibreOffice 64 ビット進行状況](#) および [「Amazon Linux 2023 のサポート」](#) を参照してください。

内容

- [前提条件](#)
- [ステップ 1: システムアーキテクチャを決定する](#)
- [ステップ 2: LibreOffice RPM パッケージをダウンロードする](#)
- [ステップ 3: アーカイブを抽出する](#)
- [ステップ 4: RPM パッケージをインストールする](#)
- [ステップ 5: インストールを確認する](#)
- [LibreOffice の起動](#)
- [トラブルシューティング](#)

前提条件

- リモートデスクトッププロトコル (Amazon EC2 DCV など) を介して設定およびアクセス可能な GNOME グラフィカルデスクトップ環境で AL2023 を実行する Amazon EC2 インスタンス。詳細については次を参照してください:

- [チュートリアル: AL2023 に GNOME デスクトップ環境をインストールする](#)
- [Amazon DCV サーバーのセットアップ](#)
- `sudo` または インスタンスのルートアクセス。
- インストール用に少なくとも 1.5 GB の空きディスク容量。
- LibreOffice パッケージをダウンロードするアクティブなインターネット接続、または事前にダウンロードされてインスタンスに転送されたパッケージ。

ステップ 1: システムアーキテクチャを決定する

次のコマンドを実行して、プロセッサアーキテクチャを識別します。これにより、ダウンロードする RPM パッケージが決まります。

```
uname -p
```

出力は次のいずれかです。

Output	アーキテクチャ	RPM パッケージタイプ
x86_64	64 ビット Intel/AMD	Linux_x86-64_rpm
aarch64	64 ビット ARM (Graviton)	Linux_aarch64_rpm

ステップ 2: LibreOffice RPM パッケージをダウンロードする

LibreOffice RPM パッケージをダウンロードするには

1. [LibreOffice のダウンロードページ](#)を開きます。
2. オペレーティングシステムとして Linux (64 ビット) (rpm) を選択します。
3. アーキテクチャに適したバージョン (x86_64 または aarch64) を選択します。
4. .tar.gz アーカイブをインスタンスにダウンロードします。wget または を使用してcurl、ターミナルから直接ダウンロードできます。

```
cd ~/Downloads
wget https://download.documentfoundation.org/libreoffice/stable/VERSION/
rpm/ARCHITECTURE/LibreOffice_VERSION_Linux_ARCHITECTURE_rpm.tar.gz
```

VERSION と **ARCHITECTURE** を、インストールするバージョン番号とアーキテクチャ (25.2.5 と x86_64 など) に置き換えます。

 Note

最新の安定バージョン番号を確認するには、[LibreOffice ダウンロードページ](#)にアクセスしてください。

ステップ 3: アーカイブを抽出する

アーカイブをダウンロードしたディレクトリに移動し、抽出します。

```
cd ~/Downloads
tar zxvf LibreOffice_VERSION_Linux_ARCHITECTURE.rpm.tar.gz
```

これにより、のような名前のディレクトリが作成されます LibreOffice_**VERSION**_Linux_**ARCHITECTURE**.rpm/。

ステップ 4: RPM パッケージをインストールする

RPM パッケージをインストールするには

1. 展開されたフォルダ内の RPMS/ サブディレクトリに変更します。

```
cd LibreOffice_VERSION_Linux_ARCHITECTURE.rpm/RPMS/
```

2. を使用してすべての RPM パッケージをインストールします `sudo rpm`。

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

または、依存関係の解決 `dnf` に を使用することもできます。

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

ステップ 5: インストールを確認する

バージョンを確認して、LibreOffice が正常にインストールされたことを確認します。

```
libreoffice --version
```

正常な出力:

```
LibreOffice VERSION ...
```

LibreOffice の起動

ターミナルから

次のコマンドを実行します。

```
libreoffice
```

libreoffice コマンドが使用できない場合は、バイナリがバージョンングされている可能性があります。代わりに、バージョンングされた名前を使用して起動できます。

```
libreofficeMAJOR_VERSION
```

例えば、libreoffice25.2。

特定のアプリケーションを直接起動するには:

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

GNOME デスクトップから

インストール後、LibreOffice アプリケーションが GNOME アクティビティメニューに表示されます。左上隅にあるアクティビティを選択し、「LibreOffice」を検索して、目的のアプリケーションを選択します。

トラブルシューティング

次のセクションは、AL2023 で LibreOffice をインストールまたは実行する際の一般的な問題のトラブルシューティングに役立ちます。

Java ランタイム環境の警告

一部の LibreOffice 機能 (基本や特定の拡張機能など) には、Java ランタイム環境 (JRE) が必要です。Java の欠落に関する警告が表示された場合は、インストールします。

```
sudo dnf install -y java-17-amazon-corretto-headless
```

次に、LibreOffice で、Tools > Options > LibreOffice > Advanced に移動し、JRE が検出されたことを確認します。

LibreOffice コマンドが見つかりません

インストール後に libreoffice コマンドが見つからない場合、バイナリはバージョンングされている可能性があります。実行してみてください。

```
libreofficeMAJOR_VERSION
```

例えば、libreoffice25.2。バイナリを見つけることもできます。

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

その他のリソース

その他の問題が発生した場合は、Linux の公式 LibreOffice インストールドキュメントを参照してください。

- [LibreOffice Linux インストールガイド](#)
- [LibreOffice ダウンロードページ](#)

Amazon EC2 以外で Amazon Linux 2023 を使用する

Amazon Linux 2023 コンテナイメージは、互換性のあるコンテナランタイム環境で実行できます。Amazon Linux 2023 をコンテナ内で使用方法の詳細については、「[コンテナでの AL2023](#)」を参照してください。

Amazon Linux 2023 (AL2023) は、Amazon EC2 で直接実行する以外に、仮想化ゲストとして実行することもできます。現在、KVM (qcow2)、VMware (OVA)、Hyper-V (vhdx) の各イメージが利用可能です。

Note

Amazon Linux 2023 イメージの設定は Amazon Linux 2 とは異なります。
[Amazon Linux 2 をオンプレミスの仮想化マシンとして実行する](#)場合は、AL2023 と互換性があるように設定を調整する必要があります。

Amazon Linux 2023 イメージをダウンロードして KVM、VMware、Hyper-V で使用する

KVM、VMware、Hyper-V で使用する Amazon Linux 2023 ディスクイメージは、cdn.amazonlinux.com からダウンロードできます。

Amazon EC2 以外の仮想化環境での使用がサポートされている Amazon Linux 2023 の設定

このセクションでは、KVM、VMware、Hyper-V などの Amazon EC2 以外の仮想化環境で Amazon Linux 2023 を実行するための要件について説明します。

基本の [AL2023 システム要件](#) は Amazon EC2 以外のすべての仮想化環境に適用されます。各ハイパーバイザー環境でサポートされるデバイスモデルのリストは、以下のトピックで詳しく説明されています。

KVM、VMware、Hyper-V には多数の設定オプションが用意されているため、セキュリティ、パフォーマンス、信頼性のニーズに合わせて設定するには注意が必要です。詳細については、ハイパーバイザーから提供されたドキュメントを参照してください。

トピック

- [KVM で AL2023 を実行するための要件](#)
- [VMware で AL2023 を実行するための要件](#)
- [Hyper-V で Amazon Linux 2023 を実行するための要件](#)

KVM で AL2023 を実行するための要件

このセクションでは、KVM で AL2023 を実行するための要件について説明します。AL2023 の KVM イメージは、aarch64 および x86-64 アーキテクチャの両方で使用できます。これらの要件は、KVM イメージの基本的な [AL2023 システム要件](#) に追加されるものです。

トピック

- [KVM で AL2023 を実行するための KVM ホスト要件](#)
- [KVM での AL2023 のデバイスサポート](#)
- [KVM での AL2023 のブートモード \(UEFI および BIOS\) サポート](#)
- [KVM で AL2023 を実行する場合の制限事項](#)

KVM で AL2023 を実行するための KVM ホスト要件

現在、KVM イメージはこの Ubuntu バージョンで提供されている qemu バージョン 6.2+dfsg-2ubuntu6.15 の Ubuntu 22.04.3 LTS を実行しているホストで、x86-64 の場合は q35 マシンタイプを、aarch64 の場合は virt マシンタイプを使用して認定されています。

KVM での AL2023 のデバイスサポート

AL2023 KVM イメージ (**aarch64** と **x86-64** の両方) での使用がテストされた **qemu** デバイスモデルは以下のとおりです。

- virtio-blk (virtio ブロックデバイス)
- virtio-scsi (ディスクデバイスを搭載した virtio SCSI コントローラ)
- virtio-net (virtio ネットワークデバイス)
- ahci (仮想化 CD-ROM ドライブ用)
- usb-storage (xhci 以上)

AL2023 KVM イメージ認証で有効にされたが、あまり実行されていないその他の **qemu** デバイスマデル

- x86-64 のみでの VGA (qemu VGA)
- virtio-rng (仮想化乱数ジェネレーター)
- 従来の AT キーボードおよび PS/2 マウスデバイス
- 従来のシリアルデバイス

KVM での AL2023 のブートモード (UEFI および BIOS) サポート

x86-64 イメージは従来の BIOS および UEFI ブートモードの両方でテストされます。aarch64 イメージは UEFI ブートモードでテストされます。

Note

デフォルトでは、UEFI ブートモードを使用すると、一部の仮想化マシンマネージャーは Microsoft セキュアブートキーを使用して VM をプロビジョニングするようになり、これによってセキュアブートが有効になります。この設定では AL2023 は起動しません。AL2023 ブートローダーは Microsoft によって署名されていないため、VM は UEFI キーなしでプロビジョニングするか、セキュアブート用の AL2023 キーを使用してプロビジョニングする必要があります。

Important

KVM イメージのセキュアブートサポートはまだ検証されていません。

KVM で AL2023 を実行する場合の制限事項

KVM で AL2023 を実行する場合には、いくつかの既知の制限があります。

Note

ここに記載されているサポートされていない機能の一部については、実装するコードが AL2023 に存在し、正しく機能する可能性があります。サポートされていない機能のリスト

は、現在使用できるものと、Amazon Linux チームが将来の更新の一環として動作を保証するものを判断するための情報として提供されています。

KVM で AL2023 を実行する場合の既知の制限事項

- KVM ゲストエージェントは現在パッケージ化されておらず、サポートもされていません。
- CPU、メモリ、またはその他の種類のデバイスのホットプラグとアンプラグはサポートされていません。
- VM の休止はサポートされていません。
- VM の移行はサポートされていません。
- PCI パススルーや USB パススルーなどによるデバイスのパススルーはサポートされていません。

VMware で AL2023 を実行するための要件

このセクションでは、VMware で AL2023 を実行するための要件について説明します。AL2023 の VMware イメージは、x86-64 アーキテクチャでのみ使用できます。aarch64 の VMware イメージは提供されておらず、サポートもされていません。これらの要件は、VMware イメージの基本的な [AL2023 システム要件](#) に追加されるものです。

トピック

- [VMware で AL2023 を実行するための VMware ホスト要件](#)
- [VMware での AL2023 のデバイスサポート](#)
- [VMware での AL2023 のブートモード \(UEFI および BIOS\) サポート](#)
- [VMware で AL2023 を実行する場合の制限事項](#)

VMware で AL2023 を実行するための VMware ホスト要件

AL2023 VMware OVA イメージは現在、以下の条件を満たしています。

- Intel(R) Xeon(R) Platinum 8124M プロセッサを搭載したホストで実行されている VMware Workstation 17.5.0
- Intel(R) Xeon(R) Platinum 8275CL プロセッサを搭載した VMware vSphere 8.0

AL2023 VMware OVA イメージでは、マシンハードウェアバージョンが 13 と指定されています。

VMware マシンハードウェアバージョン 13 は、以下によってサポートされています。

- ESXi 6.5 以降
- VMware Workstation 14 以降

VMware での AL2023 のデバイスサポート

AL2023 VMware OVA イメージでの使用がテストされている VMware デバイスモデル (x86-64 のみ)

- vmw_pvscsi (VMware 準仮想化 SCSI コントローラー)
- vmxnet3 (VMware 準仮想化ネットワークデバイス)
- ata_piix (レガシー IDE は仮想化 CD-ROM ドライブでのみ使用)

AL2023 VMware イメージ認証で有効にされたが、あまり実行されていないその他の VMware デバイスモデル

- vmw_vmci および関連の vsock インターフェイス (VMware ゲストエージェント用の仮想化ソケットトランスポート)
- vmw_balloon メモリバルーンデバイス
- VMware SVGA コントローラ
- 従来の AT キーボードおよび PS/2 マウスデバイス

VMware ゲストエージェントパッケージ (open-vm-tools) は AL2023 VMware OVA イメージにデフォルトでインストールされています。

VMware での AL2023 のブートモード (UEFI および BIOS) サポート

2023.3.20231211 リリースの時点で、AL2023 VMware OVA イメージは従来の BIOS と UEFI の両方のブートモードで検証されています。OVA のデフォルト設定は従来の BIOS のままですが、ユーザーが変更できます。

Important

セキュアブートのサポートには UEFI が必要ですが、VMware 上で実行される AL2023 に対する検証は行われていません。

VMware で AL2023 を実行する場合の制限事項

VMware で AL2023 を実行する場合には、いくつかの既知の制限があります。

Note

記載されているサポートされていない機能の一部を実装するコードが AL2023 に存在しますが、正しく機能する可能性があります。サポートされていない機能のリストは、お客様が現在の作業で何に頼るべきか、また Amazon Linux チームが今後の更新の一環として動作しているとみなすものについて、情報に基づいた決定を下せるようにするためのものです。

VMware で AL2023 を実行する場合の既知の制限事項

- UEFI セキュアブートは現在、VMware の AL2023 では検証されていません。
- CPU、メモリ、またはその他の種類のデバイスのホットプラグとアンプラグはサポートされていません。
- VM の休止はサポートされていません。
- VM の移行はサポートされていません。
- PCI パススルーや USB パススルーなどによるデバイスのパススルーはサポートされていません。

Hyper-V で Amazon Linux 2023 を実行するための要件

このセクションでは、Hyper-V で Amazon Linux 2023 を実行するための要件について説明します。AL2023 の Hyper-V イメージは、x86-64 アーキテクチャでのみ使用できます。aarch64 の Hyper-V イメージは、現時点では利用できず、サポートされていません。

このセクションでは、Hyper-V イメージの基本の[AL2023 システム要件](#)に加えて、その他の要件について説明します。

トピック

- [Hyper-V で Amazon Linux 2023 を実行するための Hyper-V ホスト要件](#)
- [Hyper-V での Amazon Linux 2023 のデバイスサポート](#)
- [Hyper-V での Amazon Linux 2023 の実行制限](#)

Hyper-V で Amazon Linux 2023 を実行するための Hyper-V ホスト要件

Hyper-V での Amazon Linux 2023 の主な認定は、EC2 の `c5.metal` インスタンスで実行している Windows Server 2022 で行われます。

Hyper-V での Amazon Linux 2023 のデバイスサポート

Amazon Linux 2023 は、以下の仮想化ハードウェアのセットを備えた第 1 世代と第 2 世代の両方の Hyper-V 仮想マシンでテストされます。

- 第 1 世代 (レガシー BIOS ブート) VM
- 第 2 世代 (UEFI ブート - セキュアブートなし) VM
- 以下のデバイスモデルは、AL2023 Hyper-V イメージでの使用がテストされています。
 - 第 2 世代 VM のルートディスクとエミュレートされた CD-ROM ドライブ用の Hyper-V 仮想ストレージ `hv_storvsc`
 - 第 1 世代 VM の仮想 CD-ROM ドライブ用のエミュレートされた PIIX IDE `ata_piix`
 - Hyper-V 仮想イーサネット `hv_netvsc`
- 以下のデバイスモデルは有効ですが、簡単にテストされています。
 - 第 1 世代 VM のレガシー VGA テキストモード
 - 第 2 世代 VM の UEFI ファームウェアベースのフレームバッファ `simplifiedrmfb`
 - Hyper-V バルーン `hv_balloon`
 - Hyper-V HID/マウス `hid_hyperv`
- 以下のデバイスモデルは、現時点において AL2023 で有効になっていません。
 - Hyper-V PCI パススルー
 - Hyper-V DRM グラフィックス

Important

第 2 第仮想マシンの場合、セキュアブートはサポートされていないため、Amazon Linux 2023 を正常に起動するには、仮想マシンを起動する前にセキュアブートを無効にする必要があります。Hyper-V は現在、Microsoft 独自のキーで署名されたソフトウェアコンポーネントを使用したセキュアブートのみをサポートしていますが、Amazon Linux ブートローダーは Amazon のプライベートキーで署名されています。Hyper-V は、現時点ではサードパーティのキーのインポートをサポートしていません。

Hyper-V での Amazon Linux 2023 の実行制限

Hyper-V で Amazon Linux 2023 を実行する場合の既知の制限は以下のとおりです。

Note

記載されているサポートされていない機能の一部を実装するコードが AL2023 に存在しますが、正しく機能する可能性があります。サポートされていない機能のリストは、お客様が現在の作業で何に頼るべきか、また Amazon Linux チームが今後の更新の一環として動作しているとみなすものについて、情報に基づいた決定を下せるようにするためのものです。

Hyper-V で AL2023 を実行する場合の既知の制限

- UEFI セキュアブートモードは現在、Hyper-V での AL2023 ではサポートされておらず、機能しません。
- CPU、メモリ、またはその他の種類のデバイスのホットプラグとアンプラグはサポートされていません。
- 仮想マシン (VM) のハイバネーションはサポートされていません。
- 仮想マシン (VM) の移行はサポートされていません。
- PCI パススルーや USB パススルーなどによるデバイスのパススルーはサポートされていません。

Amazon EC2 以外で使用する場合は Amazon Linux 2023 の設定および **cloud-init** 設定

このセクションでは、Amazon Linux 2023 仮想マシンを Amazon EC2 で直接実行せず、KVM、VMware、Hyper-V などを実行する場合にセットアップして設定する方法について説明します。

デフォルトでは、Amazon Linux 2023 の仮想マシンイメージにはユーザーパスワードや SSH キーはプロビジョニングされず、DHCP を経由して最初に検出されたネットワークインターフェイスからネットワーク設定を取得します。つまり、デフォルトでは、追加の設定を行わないと、生成された仮想化マシンに接続する方法はありません。

そのため、何らかの構成を仮想化マシンに提供する必要があります。Amazon Linux でこれを行う標準的なメカニズムは、cloud-init データソース経由です。

Amazon Linux 2023 は以下のデータソースの認定を受けています。

NoCloud

これは、cloud-init 設定ファイルを含むシード ISO9660 イメージを含む仮想化 CD-ROM を使用してオンプレミスイメージを設定する従来の方法です。

VMware

Amazon Linux 2023 では、VMware guestinfo.userdata および guestinfo.metadata 経由の VMware 固有のデータソースを介して vSphere 上で実行されている VMware イメージを設定することもサポートされています。

Note

データソースの設定は Amazon Linux 2 とは異なる場合があります。具体的には、Amazon Linux 2023 は設定に systemd-networkd を使用しており、「[cloud-init ネットワーク設定ドキュメント](#)」に記載されている cloud-init 「ネットワーク設定バージョン 2」を使用する必要があります。

Amazon Linux 2023 の cloud-init パッケージのバージョンの cloud-init 設定メカニズムに関するすべてのドキュメントは、「[cloud-init アップストリームのドキュメント](#)」にあります。

KVM および VMware での Amazon Linux 2023 の NoCloud (**seed.iso**) **cloud-init** 設定

このセクションでは、KVM または VMware で実行する Amazon Linux 2023 を設定するための seed.iso イメージを作成して使用方法について説明します。KVM 環境と VMware 環境には [Amazon EC2 インスタンスメタデータサービス \(IMDS\)](#) がないため、Amazon Linux 2023 を設定する他の方法が必要であり、seed.iso イメージの提供はその方法の 1 つです。

seed.iso ブートイメージには、ネットワーク設定、ホスト名、ユーザーデータなどの、新しい仮想化マシンの起動に必要な初期設定情報が含まれます。

Note

seed.iso イメージには、VM の起動に必要な設定情報のみ含まれています。Amazon Linux 2023 オペレーティングシステムファイルは含まれていません。

seed.iso イメージを生成するには、2 つ (もしくは 3 つ) の設定ファイルが必要です。

meta-data

このファイルには通常、仮想化マシンのホスト名が含まれます。

user-data

このファイルでは、ユーザーアカウント、パスワード、ssh キーペア、およびアクセスメカニズムを設定します。デフォルトでは、Amazon Linux 2023 KVM および VMware イメージは、ec2-user ユーザーアカウントを作成します。user-data 設定ファイルを使用して、デフォルトユーザーアカウントのパスワードおよび SSH キーを設定します。

network-config (オプション)

このファイルには通常、デフォルトのネットワーク設定に上書きされる仮想化マシンのネットワーク設定が記載されています。デフォルト設定は、最初に使用可能なネットワークインターフェースで DHCP を使用します。

seed.iso ディスクイメージを作成する

1. Linux または macOS コンピュータでは、seedconfig という名前の新しいフォルダを作成し、そのフォルダに移動します。

Note

Windows または別のオペレーティングシステムを使用してこれらのステップを実行することも可能ですが、seed.iso イメージの作成を完了するには mkisofs と同等のツールを検索する必要があります。

2. meta-data 設定ファイルを作成します。
 - a. meta-data という名前の新しいファイルを作成します。
 - b. 任意のエディタを使用して meta-data ファイルを開き、以下を追加し、*vm-hostname* を VM のホスト名に置き換えます。

```
#cloud-config
local-hostname: vm-hostname
```

- c. meta-data 設定ファイルを保存して閉じます。
3. user-data 設定ファイルを作成します。

- a. user-data という名前の新しいファイルを作成します。
- b. 任意のエディタを使用して user-data ファイルを開き、以下を追加し、必要に応じて置き換えます。

```
#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. 必要に応じて、user-data 設定ファイルにさらにユーザーアカウントを追加できます。

追加ユーザーアカウント、アクセスメカニズム、パスワード、およびキーペアを指定できます。サポートされているディレクティブの詳細については、「[アップストリーム cloud-init ドキュメント](#)」を参照してください。

- d. user-data 設定ファイルを保存して閉じます。
4. (オプション) network-config 設定ファイルを作成します。
 - a. network-config という名前の新しいファイルを作成します。
 - b. 任意のエディタを使用して network-config ファイルを開き、以下を追加し、設定に応じてさまざまな IP アドレスを IP アドレスに置き換えます。

```
#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 ネットワーク構成は、仮想化マシンの設定によって変わる可能性のあるインターフェース名を指定する代わりに、インターフェースの MAC アドレスと照合するメカニズムを提供します。ネットワーク設定用のこの (およびその他の) cloud-init 機能については、「[アップストリームの cloud-init ネットワーク設定バージョン 2 のドキュメント](#)」で詳しく説明されています。

- c. network-config 設定ファイルを保存して閉じます。
5. 前の手順で作成した meta-data、user-data、およびオプションの network-config 設定ファイルを使用して seed.iso ディスクイメージを作成します。

seed.iso ディスクイメージを作成する OS に応じて、以下のいずれかの操作を行います。

- Linux システムでは、**mkisofs** または **genisoimage** などのツールを使用して完成した seed.iso ファイルを作成します。seedconfig フォルダに移動し、以下のコマンドを実行します。

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
```

- network-config を使用する場合は、それを **mkisofs** の呼び出しに含めます。

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data  
network-config
```

- macOS システムでは、**hdiutil** などのツールを使用して完成した seed.iso ファイルを生成できます。**hdiutil** はファイルのリストではなくパス名を使用するため、network-config 設定ファイルが作成されているかどうかにかかわらず、同じ呼び出しを使用できます。

```
$ hdiutil makehybrid -o seed.iso -hfs -joliet -iso -default-volume-name cidata  
seedconfig/
```

6. 生成した seed.iso ファイルは、仮想化 CD-ROM ドライブを使用して新しい Amazon Linux 2023 仮想マシンにアタッチすることにより、cloud-init で初回起動時に設定を検索してシステムに適用できるようになります。

VMware で実行される AL2023 での VMware guestinfo の **cloud-init** 設定

VMware 環境には [Amazon EC2 インスタンスメタデータサービス \(IMDS\)](#) がないため、AL2023 を設定する別の方法が必要です。このセクションでは、VMware vSphere で使用できる `seed.iso` 仮想 CD-ROM ドライブに代わる設定メカニズムを使用する方法について説明します。

この設定方法では、VMware extraconfig メカニズムを使用して設定データを cloud-init に提供します。次の各キーには、対応する **keyname.encoding** プロパティを指定する必要があります。

VMware extraconfig メカニズムには以下のキーを指定できます。

guestinfo.metadata

cloud-init メタデータを含む JSON または YAML

guestinfo.userdata

cloud-config フォーマットの cloud-init ユーザーデータを含む YAML ドキュメント

guestinfo.vendordata (オプション)

cloud-init ベンダーデータを含む YAML

対応するエンコーディングプロパティ

(`guestinfo.metadata.encoding`、`guestinfo.userdata.encoding`、および `guestinfo.vendordata.encoding`) には以下を含めることができます。

base64

プロパティの内容は base64 エンコードされます。

gzip+base64

プロパティの内容は base64 エンコード後に gzip で圧縮されます。

Note

seed.iso メソッドは別の (オプションの) network-config 設定ファイルをサポートします。VMware guestinfo ではネットワーク設定の提供方法が異なります。以下のセクションで、さらに詳しい情報を提供しています。

明示的なネットワーク設定が必要な場合は、2 つの YAML または JSON プロパティのフォーマットで metadata に埋め込む必要があります。

network

JSON または YAML 形式でエンコードされたネットワーク設定が含まれます。

network.encoding

上記のネットワーク設定データのエンコーディングが含まれます。cloud-init でサポートされているエンコーディングは、guestinfo データのエンコーディングと共通です: base64 および gzip+base64。

Example VMware vSphere **govc** CLI ツールを使用して **guestinfo** で設定を渡す

1. 「[KVM および VMware での Amazon Linux 2023 の NoCloud \(seed.iso\) cloud-init 設定](#)」の説明に従って meta-data、user-data、およびオプションの network-config 設定ファイルを準備します。
2. 設定ファイルを VMware guestinfo で使用できる形式に変換します。

```
# '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"
```

Amazon Linux 2023 標準 AMI にインストールされているパッケージと AL2023 KVM イメージの比較

AL2023 標準 AMI の RPM と AL2023 KVM イメージの RPM の比較。

パッケージ	AMI	KVM
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1

パッケージ	AMI	KVM
amazon-onprem-netw ork		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	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28
bind-utils	9.18.28	9.18.28
binutils	2.39	2.39
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0

パッケージ	AMI	KVM
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-onpre		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

パッケージ	AMI	KVM
crypto-policies-scr ipts	20220428	20220428
cryptsetup	2.6.1	2.6.1
cryptsetup-libs	2.6.1	2.6.1
curl-minimal	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27
cyrus-sasl-plain	2.1.27	2.1.27
dbus	1.12.28	1.12.28
dbus-broker	32	32
dbus-common	1.12.28	1.12.28
dbus-libs	1.12.28	1.12.28
device-mapper	1.02.185	1.02.185
device-mapper-libs	1.02.185	1.02.185
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0
dnf-plugin-release- notification	1.2	1.2
dnf-plugins-core	4.3.0	4.3.0
dnf-plugin-support- info	1.2	1.2

パッケージ	AMI	KVM
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055
dracut-config-ec2	3.0	
dracut-config-generic	055	055
dwz	0.14	0.14
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5
e2fsprogs-libs	1.46.5	1.46.5
ec2-hibinit-agent	1.0.8	
ec2-instance-connect	1.1	
ec2-instance-connect-selinux	1.1	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5
efi-srpm-macros	5	5
efivar	38	38
efivar-libs	38	38

パッケージ	AMI	KVM
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188
elfutils-libelf	0.188	0.188
elfutils-libs	0.188	0.188
ethtool	5.15	5.15
expat	2.5.0	2.5.0
file	5.39	5.39
file-libs	5.39	5.39
filesystem	3.14	3.14
findutils	4.8.0	4.8.0
fonts-srpm-macros	2.0.5	2.0.5
fstrm	0.6.1	0.6.1
fuse-libs	2.9.9	2.9.9
gawk	5.1.0	5.1.0
gdbm-libs	1.19	1.19
gdisk	1.0.8	1.0.8
gettext	0.21	0.21
gettext-libs	0.21	0.21
ghc-srpm-macros	1.5.0	1.5.0

パッケージ	AMI	KVM
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)
grub2-pc-modules	2.06	2.06 (noarch)
grub2-tools	2.06	2.06

パッケージ	AMI	KVM
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
jemalloc	5.2.1	5.2.1
jitterentropy	3.4.1	3.4.1

パッケージ	AMI	KVM
jq	1.7.1	
json-c	0.14	0.14
kbd	2.4.0	2.4.0
kbd-misc	2.4.0	2.4.0
kernel	6.1.112	6.1.112
kernel-libbpf	6.1.112	6.1.112
kernel-livepatch-r epo-cdn		2023.6.20241031
kernel-livepatch-r epo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-ext ra-common		6.1.112
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.112	6.1.112
keyutils	1.6.3	1.6.3
keyutils-libs	1.6.3	1.6.3
kmod	29	29
kmod-libs	29	29
kpatch-runtime	0.9.7	0.9.7
krb5-libs	1.21.3	1.21.3

パッケージ	AMI	KVM
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
libdhash	0.5.0	
libdnf	0.69.0	0.69.0

パッケージ	AMI	KVM
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-hmacalc	1.4.0	1.4.0
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
libmetalink	0.1.3	0.1.3
libmnl	1.0.4	1.0.4

パッケージ	AMI	KVM
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-filessystem	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13

パッケージ	AMI	KVM
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragegmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libunistring	0.9.10	0.9.10
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

パッケージ	AMI	KVM
libverto	0.3.2	0.3.2
libverto-libev	0.3.2	0.3.2
libxcrypt	4.4.33	4.4.33
libxml2	2.10.4	2.10.4
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
linux-firmware-wheel	20210208 (noarch)	20210208 (noarch)
lm_sensors-libs	3.6.0	3.6.0
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsf	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1 (x86_64)	2.1 (x86_64)
mpfr	4.1.0	4.1.0
nano	5.8	5.8
ncurses	6.2	6.2

パッケージ	AMI	KVM
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
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1

パッケージ	AMI	KVM
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1
openssl	3.0.8	3.0.8
openssl-lib	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-lib	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50
perl-Class-Struct	0.66	0.66
perl-constant	1.33	1.33
perl-DynaLoader	1.47	1.47

パッケージ	AMI	KVM
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
perl-overload	1.31	1.31
perl-overloading	0.02	0.02

パッケージ	AMI	KVM
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
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300

パッケージ	AMI	KVM
perl-vars	1.05	1.05
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0
polycoreutils	3.4	3.4
polycoreutils-python-utils	3.4	
popt	1.18	1.18
procps-ng	3.3.17	3.3.17
protobuf-c	1.4.1	1.4.1
psacct	6.6.4	6.6.4
psmisc	23.4	23.4
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-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

パッケージ	AMI	KVM
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-jsonschemata	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0

パッケージ	AMI	KVM
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
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1

パッケージ	AMI	KVM
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
rng-tools	6.14	6.14
rootfiles	8.1	8.1

パッケージ	AMI	KVM
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
sqlite-libs	3.40.0	3.40.0
sssd-client	2.9.4	2.9.4

パッケージ	AMI	KVM
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
tcsh	6.24.07	6.24.07
time	1.9	1.9

パッケージ	AMI	KVM
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
xfspgrog	5.18.0	5.18.0
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0
xz	5.2.5	5.2.5

パッケージ	AMI	KVM
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-def aults	1.1.2	
zstd	1.5.5	1.5.5

Amazon Linux 2023 標準 AMI と AL2023 VMware OVA イメージ間でのインストールされているパッケージの比較

AL2023 標準 AMI の RPM と AL2023 VMware OVA イメージの RPM の比較。

パッケージ	AMI	VMware OVA
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo- cdn		2023.6.20241031

パッケージ	AMI	VMware OVA
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-netw ork		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	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.18.28	9.18.28
bind-license	9.18.28	9.18.28
bind-utils	9.18.28	9.18.28
binutils	2.39	2.39

パッケージ	AMI	VMware OVA
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-onp rem		22.2.2
cloud-utils-growpart	0.31	0.31
coreutils	8.32	8.32
coreutils-common	8.32	8.32
cpio	2.13	2.13
cracklib	2.9.6	2.9.6
cracklib-dicts	2.9.6	2.9.6
crontabs	1.11	1.11

パッケージ	AMI	VMware OVA
crypto-policies	20220428	20220428
crypto-policies-scripts	20220428	20220428
cryptsetup	2.6.1	2.6.1
cryptsetup-libs	2.6.1	2.6.1
curl-minimal	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27
cyrus-sasl-plain	2.1.27	2.1.27
dbus	1.12.28	1.12.28
dbus-broker	32	32
dbus-common	1.12.28	1.12.28
dbus-libs	1.12.28	1.12.28
device-mapper	1.02.185	1.02.185
device-mapper-libs	1.02.185	1.02.185
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0
dnf-plugin-release-notification	1.2	1.2
dnf-plugins-core	4.3.0	4.3.0

パッケージ	AMI	VMware OVA
dnf-plugin-support-info	1.2	1.2
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055
dracut-config-ec2	3.0	
dracut-config-generic	055	055
dwz	0.14	0.14
dyninst	10.2.1	10.2.1
e2fsprogs	1.46.5	1.46.5
e2fsprogs-libs	1.46.5	1.46.5
ec2-hibinit-agent	1.0.8	
ec2-instance-connect	1.1	
ec2-instance-connect-selinux	1.1	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5
efi-srpm-macros	5	5
efivar	38	38

パッケージ	AMI	VMware OVA
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
gdbm-libs	1.19	1.19

パッケージ	AMI	VMware OVA
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
grub2-efi-x64-ec2	2.06	2.06

パッケージ	AMI	VMware OVA
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
irqbalance	1.9.0	1.9.0

パッケージ	AMI	VMware OVA
jansson	2.14	2.14
jemalloc	5.2.1	5.2.1
jitterentropy	3.4.1	3.4.1
jq	1.7.1	
json-c	0.14	0.14
kbd	2.4.0	2.4.0
kbd-misc	2.4.0	2.4.0
kernel	6.1.112	6.1.112
kernel-libbpf	6.1.112	6.1.112
kernel-livepatch-r epo-cdn		2023.6.20241031
kernel-livepatch-r epo-s3	2023.6.20241031	
kernel-modules-extra		6.1.112
kernel-modules-ext ra-common		6.1.112
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.112	6.1.112
keyutils	1.6.3	1.6.3
keyutils-libs	1.6.3	1.6.3
kmod	29	29

パッケージ	AMI	VMware OVA
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
libcurl-minimal	8.5.0	8.5.0

パッケージ	AMI	VMware OVA
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-hmacalc	1.4.0	1.4.0
libldb	2.6.2	

パッケージ	AMI	VMware OVA
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-fs	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4

パッケージ	AMI	VMware OVA
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragegmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libtool-ltdl		2.4.7

パッケージ	AMI	VMware OVA
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
lz4-libs	1.9.4	1.9.4

パッケージ	AMI	VMware OVA
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
ntsysv	1.15	1.15

パッケージ	AMI	VMware OVA
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
pciutils	3.7.0	3.7.0

パッケージ	AMI	VMware OVA
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
perl-interpreter	5.32.1	5.32.1

パッケージ	AMI	VMware OVA
perl-I0	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21

パッケージ	AMI	VMware OVA
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
publicsuffix-list-dafsa	20240212	20240212

パッケージ	AMI	VMware OVA
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
python3-hawkey	0.69.0	0.69.0
python3-idna	2.10	2.10

パッケージ	AMI	VMware OVA
python3-jinja2	2.11.3	2.11.3
python3-jmespath	0.10.0	0.10.0
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
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

パッケージ	AMI	VMware OVA
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-setuptools	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

パッケージ	AMI	VMware OVA
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
sed	4.8	4.8
selinux-policy	38.1.45	38.1.45

パッケージ	AMI	VMware OVA
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
systemd-udev	252.23	252.23
system-release	2023.6.20241031	2023.6.20241031

パッケージ	AMI	VMware OVA
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
wget	1.21.3	1.21.3
which	2.21	2.21

パッケージ	AMI	VMware OVA
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfspgrog	5.18.0	5.18.0
xmlsec1		1.2.33
xmlsec1-openssl		1.2.33
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0
xz	5.2.5	5.2.5
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-def aults	1.1.2	
zstd	1.5.5	1.5.5

Amazon Linux 2023 標準 AMI と AL2023 Hyper-V イメージ間でのインストールされているパッケージの比較

AL2023 標準 AMI の RPM と AL2023 Hyper-V イメージの RPM の比較。

パッケージ	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-netw ork		1.2
amazon-rpm-config	228	228
amazon-ssm-agent	3.2.2303.0	3.2.2303.0
at	3.1.23	3.1.23
attr	2.5.1	2.5.1
audit	3.0.6	3.0.6
audit-libs	3.0.6	3.0.6
aws-cfn-bootstrap	2.0	
awscli-2	2.14.5	2.14.5
basesystem	11	11

パッケージ	AMI	Hyper-V VHDX
bash	5.2.15	5.2.15
bash-completion	2.11	2.11
bc	1.07.1	1.07.1
bind-libs	9.16.48	9.16.48
bind-license	9.16.48	9.16.48
bind-utils	9.16.48	9.16.48
binutils	2.39	2.39
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0
boost-thread	1.75.0	1.75.0
bzip2	1.0.8	1.0.8
bzip2-libs	1.0.8	1.0.8
ca-certificates	2023.2.64	2023.2.64
c-ares	1.19.0	
checkpolicy	3.4	3.4
chkconfig	1.15	1.15
chrony	4.3	4.3
cloud-init	22.2.2	22.2.2
cloud-init-cfg-ec2	22.2.2	
cloud-init-cfg-onpre		22.2.2

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
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	

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
hwdata	0.353	0.353
hyperv-daemons		0
hyperv-daemons-lic ense		0
hypervfcopyd		0
hypervkvpd		0

パッケージ	AMI	Hyper-V VHDX
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
kernel-livepatch-r epo-cdn		2023.4.20240319
kernel-livepatch-r epo-s3	2023.4.20240319	
kernel-modules-extra		6.1.79

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
libcbor	0.7.0	0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.46.5
libcomps	0.1.20	0.1.20
libconfig	1.7.2	1.7.2
libcurl-minimal	8.5.0	8.5.0
libdb	5.3.28	5.3.28
libdhash	0.5.0	
libdnf	0.69.0	0.69.0
libeconf	0.4.0	0.4.0
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12
libfdisk	2.37.4	2.37.4
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragegmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-softhoken	3.90.0	3.90.0
nss-softhoken-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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

パッケージ	AMI	Hyper-V VHDX
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

パッケージ	AMI	Hyper-V VHDX
perl-Storable	3.21	3.21
perl-subvars	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

パッケージ	AMI	Hyper-V VHDX
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-awscli	0.19.19	0.19.19
python3-babel	2.9.1	2.9.1
python3-cffi	1.14.5	1.14.5
python3-chardet	4.0.0	4.0.0
python3-colorama	0.4.4	0.4.4
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

パッケージ	AMI	Hyper-V VHDX
python3-hawkey	0.69.0	0.69.0
python3-idna	2.10	2.10
python3-jinja2	2.11.3	2.11.3
python3-jmespath	0.10.0	0.10.0
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11

パッケージ	AMI	Hyper-V VHDX
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-too lkit	3.0.24	3.0.24
python3-pycparser	2.20	2.20
python3-pyrsistent	0.17.3	0.17.3
python3-pyserial	3.4	3.4
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1
python3-pyyaml	5.4.1	5.4.1
python3-requests	2.25.1	2.25.1
python3-rpm	4.16.1.3	4.16.1.3
python3-ruamel-yaml	0.16.6	0.16.6
python3-ruamel-yaml- clib	0.1.2	0.1.2
python3-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

パッケージ	AMI	Hyper-V VHDX
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

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
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
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

パッケージ	AMI	Hyper-V VHDX
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfspgrog	5.18.0	5.18.0
xxd	9.0.2153	9.0.2153
xxhash-libs	0.8.0	0.8.0
xz	5.2.5	5.2.5
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-def aults	1.1.2	
zstd	1.5.5	1.5.5

Amazon Linux のインスタンスとバージョンの識別

OS イメージまたはインスタンスが、どの Linux ディストリビューションで、どのバージョンのディストリビューションであるかを判別することが重要になる場合があります。Amazon Linux には、Amazon Linux を他の Linux ディストリビューションから識別し、イメージがどのリリースの Amazon Linux であるかを判別するメカニズムが備わっています。

このセクションでは、使用できるさまざまな方法、各方法の制限、いくつかの使用例を紹介します。

トピック

- [os-release 標準の使用](#)
- [Amazon Linux 固有](#)
- [OS 検出のコード例](#)

os-release 標準の使用

Amazon Linux は、Linux ディストリビューションを識別するための [os-release 標準](#) に準拠しています。このファイルは、オペレーティングシステムの識別とバージョンに関する機械読み取り可能な情報を提供します。

Note

標準では、最初に `/etc/os-release` を解析し、次に `/usr/lib/os-release` を解析するよう規定しています。ファイル名とパスに関しては、標準に従うよう注意する必要があります。

トピック

- [識別の主な違い](#)
- [フィールドタイプ: 機械読み取り可能なタイプと人間が読み取り可能なタイプの比較](#)
- [/etc/os-release の例](#)
- [他のディストリビューションとの比較](#)

識別の主な違い

os-release は /etc/os-release にあります。そこにはない場合は /usr/lib/os-release にあります。詳細については、[os-release 標準](#)を参照してください。

インスタンスが Amazon Linux を実行しているかどうかを判断する最も信頼性の高い方法は、os-release の ID フィールドを確認することです。

バージョンを区別する最も信頼性の高い方法は、os-release の VERSION_ID フィールドを確認することです。

- Amazon Linux AMI: VERSION_ID は日付ベースのバージョン (例: 2018.03) を示します。
- AL2: VERSION_ID="2"
- AL2023: VERSION_ID="2023"

Note

VERSION_ID はプログラムによる使用を目的とした機械読み取り可能なフィールドであり、PRETTY_NAME はユーザーへの表示用に設計されていることに注意してください。フィールドタイプの詳細については、「[the section called “フィールドタイプ”](#)」を参照してください。

フィールドタイプ: 機械読み取り可能なタイプと人間が読み取り可能なタイプの比較

/etc/os-release ファイル (/etc/os-release が存在しない場合は /usr/lib/os-release) には、プログラムによる使用を目的とした機械読み取り可能なフィールドと、ユーザーへの表示を目的とした人間が読み取り可能なフィールドという 2 つのタイプのフィールドが含まれています。

機械読み取り可能なフィールド

標準化された形式を使用するフィールドであり、スクリプト、パッケージマネージャー、その他の自動ツールによる処理を目的としています。小文字、数字、限定された句読点 (ピリオド、アンダースコア、ハイフン) のみを使用できます。

- ID – オペレーティングシステム識別子。Amazon Linux はすべてのバージョンで `amzn` を使用し、Debian (`debian`)、Ubuntu (`ubuntu`)、Fedora (`fedora`) などの他のディストリビューションと区別しています。
- VERSION_ID – プログラムで使用するオペレーティングシステムのバージョン (例: 2023)
- ID_LIKE – 関連するディストリビューションのスペース区切りリスト (例: `fedora`)
- VERSION_CODENAME – スクリプトのリリースコード名 (例: `karoo`)
- VARIANT_ID – プログラムで決定するバリエーション識別子
- BUILD_ID – システムイメージのビルド識別子
- IMAGE_ID – コンテナ化された環境のイメージ識別子
- PLATFORM_ID – プラットフォーム識別子 (例: `platform:al2023`)

人間が読み取り可能なフィールド

ユーザーへの表示を目的としたフィールドであり、スペース、大文字と小文字の混在、説明テキストを使用できます。ユーザーインターフェイスでオペレーティングシステム情報を表示するときに使用してください。

- NAME – 表示用のオペレーティングシステム名 (例: `Amazon Linux`)
- PRETTY_NAME – 表示用のバージョンを含む完全なオペレーティングシステム名 (例: `Amazon Linux 2023.8.20250721`)
- VERSION – ユーザーへの表示に適したバージョン情報
- VARIANT – 表示用のバリエーション名またはエディション名 (例: `Server Edition`)

その他の情報フィールド

オペレーティングシステムに関する追加のメタデータを示すフィールドです。

- HOME_URL – プロジェクトのホームページ URL
- DOCUMENTATION_URL – ドキュメント URL
- SUPPORT_URL – サポート情報 URL
- BUG_REPORT_URL – バグレポート URL
- VENDOR_NAME – ベンダー名
- VENDOR_URL – ベンダー URL
- SUPPORT_END – サポート終了日 (YYYY-MM-DD 形式)

- CPE_NAME – 共通プラットフォーム列挙識別子
- ANSI_COLOR – ターミナル表示用の ANSI カラーコード

Amazon Linux をプログラムで識別する必要があるスクリプトやアプリケーションを作成するときは、ID や VERSION_ID などの機械読み取り可能なフィールドを使用します。ユーザーにオペレーティングシステム情報を表示するときは、PRETTY_NAME などの人間が読み取り可能なフィールドを使用します。

/etc/os-release の例

/etc/os-release ファイルの内容は Amazon Linux のバージョンによって異なります。

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/"
```

他のディストリビューションとの比較

Amazon Linux が広範な Linux エコシステムにどのように適合するかを理解するには、その `/etc/os-release` 形式を他の主要なディストリビューションと比較します。

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
```

機械読み取り可能なフィールドは、ディストリビューション間を一貫した方法で識別することに注目してください。

- ID – オペレーティングシステムを一意に識別: Amazon Linux 用の `amzn`、Fedora 用の `fedora`、Debian 用の `debian`、Ubuntu 用の `ubuntu`
- ID_LIKE – ディストリビューションの関係を表示: Amazon Linux は `fedora` (AL2023) または `centos` `rhel` `fedora` (AL2) を使用、Ubuntu は Debian の伝統を示す `debian` を表示
- VERSION_ID – 機械解析可能なバージョン情報を提供: AL2023 用の `2023`、Fedora 用の `42`、Debian 用の `12`、Ubuntu 用の `24.04`

対照的に、人間が読み取り可能なフィールドはユーザーへの表示用に設計されています。

- NAME – ユーザーフレンドリーな OS 名: Amazon Linux、Fedora Linux、Debian GNU/Linux、Ubuntu
- PRETTY_NAME – 完全な表示名とバージョン: Amazon Linux 2023.8.20250721、Fedora Linux 42 (Container Image)、Debian GNU/Linux 12 (bookworm)、Ubuntu 24.04.2 LTS
- VERSION – コード名やリリースタイプなどの追加のコンテキストを含む、人間が読み取り可能なバージョン

クロスプラットフォームスクリプトを記述する場合、ロジックと決定には常に機械読み取り可能なフィールド (ID、VERSION_ID、ID_LIKE) を使用し、人間が読み取り可能なフィールド (PRETTY_NAME、NAME) はユーザーに情報を表示する目的にのみ使用します。

Amazon Linux 固有

一部のファイルは Amazon Linux 固有であり、Amazon Linux とそのバージョンを識別するために使用できます。新しいコードでは、クロスディストリビューションの互換性を確保するために、[/etc/os-release](#) 標準を使用する必要があります。Amazon Linux 固有のファイルの使用はお勧めしません。

トピック

- [/etc/system-release ファイル](#)
- [イメージ識別ファイル](#)
- [Amazon Linux 固有のファイルの例](#)

/etc/system-release ファイル

Amazon Linux には、インストールされている現在のリリースを示す `/etc/system-release` ファイルが含まれています。このファイルはパッケージマネージャーを使用して更新され、Amazon Linux では `system-release` パッケージの一部となっています。このファイルは、Fedora などの他のディストリビューションにもありますが、Ubuntu などの Debian ベースのディストリビューションにはありません。

Note

`/etc/system-release` ファイルには人間が読み取れる文字列が含まれているため、OS またはリリースを識別するためにプログラムで使用しないでください。代わりに、`/etc/os-release` (`/etc/os-release` が存在しない場合は `/usr/lib/os-release`) で機械読み取り可能なフィールドを使用してください。

Amazon Linux には、`/etc/system-release-cpe` ファイルの Common Platform Enumeration (CPE) 仕様に準拠した `/etc/system-release` の機械読み取り可能なバージョンも含まれています。

イメージ識別ファイル

各 Amazon Linux イメージには、Amazon Linux チームが生成した元のイメージに関する追加情報を示す独自の `/etc/image-id` ファイルが含まれています。このファイルは Amazon Linux に固有であり、Debian、Ubuntu、Fedora などの他の Linux ディストリビューションにはありません。このファイルには、イメージに関する次の情報が含まれています。

- `image_name`、`image_version`、`image_arch` - イメージの構築に使用したビルドレシピからの値。
- `image_stamp` - イメージの作成中に生成される一意のランダムな 16 進値。
- `image_date` - YYYYMMDDhhmmss 形式で画像を作成した UTC 時間。

- `recipe_name`、`recipe_id` – イメージの構築に使用したビルドレシピの名前と ID。

Amazon Linux 固有のファイルの例

以下のセクションでは、Amazon Linux のメジャーバージョンごとに Amazon Linux 固有の識別ファイルの例を示します。

Note

実際のコードでは、`/etc/os-release` ファイルが存在しない場合、`/usr/lib/os-release` を使用する必要があります。

AL2023

以下の例は、AL2023 の識別ファイルを示しています。

AL2023 の `/etc/image-id` の例:

```
[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 の `/etc/system-release` の例:

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

```
Amazon Linux release 2023.8.20250721 (Amazon Linux)
```

AL2

以下の例は、AL2 の識別ファイルを示しています。

AL2 の /etc/image-id の例:

```
[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 の /etc/system-release の例:

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

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

Amazon Linux AMI

以下の例は、Amazon Linux AMI の識別ファイルを示しています。

Amazon Linux AMI の /etc/image-id の例:

```
[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 の /etc/system-release の例:

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

Amazon Linux AMI release 2018.03

OS 検出のコード例

以下の例は、`/etc/os-release` (`/etc/os-release` が存在しない場合は `/usr/lib/os-release`) ファイルを使用してオペレーティングシステムとバージョンをプログラムで検出する方法を示しています。これらの例は、Amazon Linux と他のディストリビューションを区別する方法と、ID_LIKE フィールドを使用してディストリビューションファミリーを決定する方法を示しています。

次のスクリプトは、複数の異なるプログラミング言語で実装され、各実装が同じ出力を生成します。

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";
    }
}

```

```
}
```

異なるシステムで実行すると、スクリプトは次の出力を生成します。

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)
```

ファイルシステムのレイアウト

このセクションでは、インスタンスまたは AL2023 ベースのコンテナに固有の詳細を含め、AL2023 システムのファイルシステムのレイアウトについて説明します。詳細については、`file-hierarchy(7) man` ページを参照してください。

トピック

- [/\(ルートディレクトリ\)](#)
- [/boot \(カーネル、initramfs など\)](#)
- [/etc \(システム設定\)](#)
- [/home \(ユーザーのホームディレクトリ\)](#)
- [/root \(root ユーザーのホームディレクトリ\)](#)
- [/srv \(サーバーバイロード\)](#)
- [/tmp \(小さな一時ファイル\)](#)
- [/run \(ランタイムデータ\)](#)
- [/usr \(システムリソース\)](#)
- [/var \(永続的な可変システムデータ\)](#)

/(ルートディレクトリ)

デフォルトでは、AL2023 イメージは書き込み可能な / を使用して設定され、特権ユーザーは新しいファイルとディレクトリを作成できます。

別のパスまたはイメージを使用するように systemd サービスを設定し、そのサービスの / として表示することや、任意のパスにアクセス制限を設定することができます。

Note

サービスのアクセス対象を制限するように systemd サービスを設定するのがベストプラクティスです。これには、そのサービスの / を読み取り専用にする `ReadOnlyPaths=/` ディレクティブの使用が含まれます。

systemd を使用してサービスからシステムへのアクセスを制限する方法の詳細については、`systemd.exec(5) man` ページを参照してください。

/boot (カーネル、initramfs など)

デフォルトでは、起動可能な AL2023 イメージは、root ファイルシステムに配置されるように /boot で設定されています。/boot パスはブート可能なイメージにのみ関連するため、AL2023 コンテナイメージでは使用されません。

このディレクトリには、Linux カーネルや initramfs など、AL2023 を起動するために必要なファイルがあります。このディレクトリのコンテンツは、OS に付属のツールのみを使用して操作する必要があります。

/boot/efi (EFI システムパーティション)

デフォルトでは、起動可能な AL2023 イメージは、/boot/efi にマウントされている EFI System パーティションで設定されます。このファイルシステムは OS によって管理され、システムの起動に不可欠なコードと設定を含んでいます。

このパスはコンテナイメージには関係ありません。

/etc (システム設定)

AL2023 の /etc ディレクトリには、システム固有の設定が含まれています。デフォルトでは、AL2023 イメージは root ファイルシステムの /etc に付属しており、特権ユーザーによって書き込み可能です。

Note

アプリケーション (systemd を含む) では、デフォルト設定を [/usr \(システムリソース\)](#) 保持することが一般的であり、設定を [/etc \(システム設定\)](#) に配置することで上書きが可能になります。

これらのアプリケーションでは、/etc のデフォルト設定を上書きせずに、[/usr \(システムリソース\)](#) のファイルを変更すると、パッケージの更新時に変更が上書きされる可能性があります。

/home (ユーザーのホームディレクトリ)

通常のユーザーでは /home の下にホームディレクトリがありますが、ソフトウェアは /home/\$USER などのパターンに依存せずに、常にユーザーごとの \$HOME 環境変数を探すはずです。

デフォルトでは、AL2023 イメージには root ファイルシステムに /home がありますが、ソフトウェアはこれに依存すべきではありません。OS の設定として /home を独立したファイルシステムとして構成し、起動の後半やユーザーの認証後にシステムにマウントするのは有効なやり方です。

root ユーザーのホームディレクトリは /home ではなく [/root \(root ユーザーのホームディレクトリ\)](#) にあるため、/home ファイルシステムをマウントできない場合に使用できます。

Note

/home への書き込みアクセスが不要な systemd サービスでは、ProtectHome=read-only ディレクティブを使用した設定がベストプラクティスです。このオプションでは、/home、/root、および /run/user は サービスに対して読み取り専用になります。

また、/home へのアクセスを必要としないサービスについては、ProtectHome=tmpfs ディレクティブで設定するのがベストプラクティスです。これによりサービスはサンドボックス内で実行され、その中では /home、/root、/run/user が空の読み取り専用の tmpfs ファイルシステムとして扱われます。

systemd を使用してサービスからシステムへのアクセスを制限する方法の詳細については、systemd.exec(5) man ページを参照してください。

/root (root ユーザーのホームディレクトリ)

root ユーザーのホームディレクトリは /root ディレクトリにあり、[/home \(ユーザーのホームディレクトリ\)](#) とは意図的に分離されているため、[/home \(ユーザーのホームディレクトリ\)](#) が利用できないファイルシステム上にある場合でも利用できます。

systemd サービス設定のベストプラクティスは、/root の場合も [/home \(ユーザーのホームディレクトリ\)](#) の場合も同じです。

/srv (サーバーペイロード)

/srv ディレクトリはシステムの管理者が管理します。Amazon Linux 2023 では、このディレクトリの整理方法に制限はありません。

/srv ディレクトリを別のファイルシステム上に設定でき、その場合は起動の後半にのみ使用可能になる可能性があります。

/tmp (小さな一時ファイル)

Note

Amazon Linux 2023 は Amazon Linux 2 とは異なり、デフォルトでは /tmp は root ファイルシステムのパスではなく、tmpfs になります。

Note

コンテナで実行する場合、通常、/tmp が tmpfs であるか、ディスク上のパスであるか、または実行中のクリーンアッププロセスがあるかどうかを指定するコンテナランタイム設定になります。

/tmp ディレクトリは、サイズが制限された小さな一時ファイル用です。デフォルトでは、AL2023 はこのディレクトリをサイズ制限が RAM の 50%、inodes が最大 100 万の tmpfs ファイルシステムとして設定します。

アプリケーションは、/tmp よりも \$TMPDIR 環境変数のパスを優先する必要があります。そうすることで、ユーザーは \$TMPDIR 環境変数を設定して、アプリケーションが /tmp に使用するパスを上書きできます。

大きな一時ファイルの場合は、代わりに [/var/tmp](#) を使用する必要があります。

Warning

/tmp は共有されるため、一時ファイルの作成に安全な方法を使用することが重要です。詳細については、「[/tmp と /var/tmp の安全な使用](#)」に関するアップストリーム systemd ドキュメントを参照してください。

Note

systemd サービスについては、PrivateTmp= デイレクティブを yes または disconnected に設定して構成するのがベストプラクティスです。これによりサービスはサンドボックス内で実行され、その中では /tmp と [/var/tmp](#) はホストや他のサービスとは共有されません。

同じプライベート一時ディレクトリを共有するように 2 つのサービスを設定する方法などの詳細については、`systemd.exec(5)` man ページを参照してください。

`/tmp` のコンテンツは通常、起動時にクリーンアップされ、未使用のファイルは定期的にクリーンアップされます。デフォルトでは、クリーンアッププロセスは起動直後に実行され、その後毎日実行されます。一時ファイルのクリーンアップを設定する方法については、`tmpfiles.d(5)` および `systemd-tmpfiles(8)` man マニュアルページを参照してください。

`/tmp` パスと [/var/tmp](#) パスは密接に関連しており、異なる目的のために存在します。

`/run` (ランタイムデータ)

`/run` ディレクトリは、システムパッケージが小さなランタイムデータ (ソケットファイルなど) を保存するために使用されます。これは `tmpfs` ファイルシステムであり、特権プログラムによってのみ書き込み可能です。

`/run/log` ディレクトリは、システムコンポーネントがログを保存するために使用できます。これは、ログが `/var/log` に書き出される前、または `/var/log` ファイルシステムが使用可能になる前の段階で使用されます。

`/run/user/` パスには、ユーザーごとのランタイムディレクトリが含まれます。デフォルトでは、これらは個別の `tmpfs` ファイルシステムとして構成され、ユーザーがログインすると `systemd` によってマウントされ、ログアウトすると消去されます。[XDG ベースディレクトリ仕様](#)に従って、これらのパスは直接参照するのではなく、`$XDG_RUNTIME_DIR` 環境変数を介して参照する必要があります。

`/usr` (システムリソース)

`/usr` 階層は、ベンダーが提供するオペレーティングシステムリソース用です。[/usr/local](#) 階層を除き、OS パッケージマネージャー以外は、`/usr` 下では一切変更しないでください。

ソフトウェアアプリケーションは、`/usr` が読み取り専用であることを前提とする必要があります。`/usr` 階層を変動データに使用しないでください。[/usr/local](#) を除き、OS パッケージマネージャーによるパッケージのインストール/削除の以外で追加または変更されるデータに `/usr` 階層を使用しないでください。OS パッケージマネージャーは、すべての `/usr` 階層 ([/usr/local](#) を除く) が同じマウントポイントであると想定する場合があります。

OS パッケージマネージャーの外部にインストールされるソフトウェアは、OS パッケージマネージャーの今後の呼び出しを妨げる可能性があるため、`/usr` にデータを保存しないでください。[`/usr/local`](#) 階層は例外であり、OS パッケージマネージャー以外のソフトウェア用に予約されています。

`/usr/bin` (実行可能ファイル)

標準的な検索 `$PATH` に表示されるべき実行可能ファイル。シェルから呼び出す場合に便利です。シェルから呼び出す必要がないデーモンと実行可能ファイルは、代わりに `/usr/lib` または `/usr/libexec` に置かれます。

`/usr/include` (C/C++ ヘッダー)

`/usr/include` ディレクトリには C および C++ ヘッダーファイルが含まれています。通常は `-devel` サフィックスが付いたパッケージに含まれています。

`/usr/lib` および `/usr/lib64` (共有ライブラリ)

Amazon Linux 2023 では、`/usr/lib64` パスは 64 ビットの共有ライブラリと、アーキテクチャに依存するパッケージデータに使用されます。AL2023 には 32 ビットのユーザースペースサポートが付属していないため、利用できる共有ライブラリは 64 ビットのみです。

`/usr/lib` パスは、すべてのアーキテクチャと互換性のある OS パッケージからの静的データ用です。これには、通常シェルから呼び出されない実行可能ファイルが含まれる場合があります。これは、`/usr/libexec` でも確認できます。共有ライブラリは、`/usr/lib64` ではなく `/usr/lib` にあります。

`/usr/local` (システム管理者がインストールしたソフトウェア)

Amazon Linux 2023 では、システム管理者が OS が管理していないソフトウェアをインストールするために `/usr/local` パスを使用できます。OS により変更されることはありません。デフォルトの `/usr/local` 階層は `/` 階層をミラーリングします。

`/usr/share` (共有リソース)

ドキュメント、フォント、タイムゾーンデータなどの共有リソースは `/usr/share` に置かれます。このディレクトリでは、さまざまな仕方でデータを保存する場所と形式を正確に指定することが一般的です。

/usr/share/doc (ドキュメント)

パッケージに付属するドキュメントは `/usr/share/doc` に保存されます。

/var (永続的な可変システムデータ)

/var/cache (キャッシュ)

[/var/lib](#) とは対照的に、`/var/cache` でデータを消去してもデータが失われることはありません。アプリケーションは他のソースから `/var/cache` データを再構築する必要があるためです。

/var/lib (永続的なシステムデータ)

`/var/lib` ディレクトリは永続的なシステムデータに使用されます。さまざまなシステムコンポーネントがそのコンポーネント専用のデータをここに配置します。[/var/cache](#) とは対照的に、`/var/lib` のデータを消去すると、データが失われます。

例えば、PostgreSQL データベースサーバーはデータベースデータをデフォルトで `/var/lib/pgsql` に保存します。このデータのレイアウトとファイル形式は PostgreSQL 固有のものです。また、永続的なデータであり、消去されるとユーザーはデータを失います。

/var/log (永続ログ)

このディレクトリは永続ログの保存に使用されます。ログファイルを `/var/log` に直接保存するのではなく、ソフトウェアで `syslog(3)` または `sd_journal_print(3)` API コールを使用することをお勧めします。

Note

AL2023 では[systemd ジャーナルによる rsyslog の置き換え](#) はデフォルトの Amazon Linux 2 設定と大きく異なります。

`journalctl` を使用したログの読み取りの詳細については、[journalctl](#) のマニュアルページを参照してください。

多くのアプリケーションは、`/var/log` にあるログファイルを記述したり、ローテーションしたりするために独自の仕組みを使用します。ログファイルの設定方法については、これらのアプリケーションのドキュメントを参照してください。

/var/spool (メールキューとプリンターキュー)

このディレクトリは、メールキューやプリンターキューなどの永続データに使用されます。

/var/tmp (より大きな一時ファイル)

サイズが小さい一時ファイルの場合は、代わりに [/tmp](#) を使用する必要があります。

[/tmp](#) はデフォルトで tmpfs ボリュームとして設定されていますが、`/var/tmp` はデフォルトでルートファイルシステムのパスとして定されています。したがって、より大きく永続的な一時ファイルの置き場所となります。デフォルトでは、クリーンアップジョブが定期的に行われ、最近アクセスされていないファイルを削除します。

一時ファイルのクリーンアップを設定する方法については、`tmpfiles.d(5)` および `systemd-tmpfiles(8) man` マニュアルページを参照してください。

[/tmp](#) と同様に、アプリケーションは `/var/tmp` よりも `$TMPDIR` 環境変数で指定されたパスを優先する必要があります。そうすることで、ユーザーは `$TMPDIR` 環境変数を設定して、アプリケーションが `/var/tmp` に使用するパスを上書きできます。

Warning

`/var/tmp` は共有されているため ([/tmp](#) と同様に)、一時ファイルの作成には安全な方法を使用することが重要です。詳細については、「[/tmp と /var/tmp の安全な使用](#)」に関するアップストリーム `systemd` ドキュメントを参照してください。

Note

`systemd` サービスについては、`PrivateTmp=` ディレクティブを `yes` または `disconnected` に設定して構成するのがベストプラクティスです。これによりサービスはサンドボックス内で実行され、その中では [/tmp](#) と [/var/tmp](#) はホストや他のサービスとは共有されません。

同じプライベート一時ディレクトリを共有するように 2 つのサービスを設定する方法などの詳細については、`systemd.exec(5) man` ページを参照してください。

[/tmp](#) パスと [/var/tmp](#) パスは密接に関連しており、異なる目的のために存在します。

AL2023 の更新

セキュリティ更新プログラムや新機能を活用するには、AL2023 リリースを常に最新の状態に保つことが重要です。AL2023 を使用すると、[AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)を使用して、環境全体でパッケージバージョンおよび更新の一貫性を確保できます。

Warning

`dnf --releasever=latest update` を実行することはベストプラクティスではありません。これを実行すると、OS の更新が最初に本番環境でテストされる結果を招く可能性があります。

`latest` を使用する代わりに、特定の AL2023 リリースバージョンを使用してください。これにより、以前にテストしたのと同じ変更を本稼働インスタンスにデプロイできます。たとえば、`dnf --releasever=2023.11.20260526 update` は常に 2023.11.20260526 リリースに更新されます。

詳細については、「[AL2023 ユーザーガイド](#)」の「[AL2023 の更新](#)」を参照してください。

トピック

- [更新を安全にデプロイするためのベストプラクティス](#)
- [新しいアップデートの通知を受け取る](#)
- [AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)
- [AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)
- [AL2023 のカーネルライブパッチ](#)
- [AL2023 での Linux カーネルの更新](#)

更新を安全にデプロイするためのベストプラクティス

Amazon Linux 2023 (AL2023) には、オペレーティングシステムの更新を安全にデプロイし、更新間で何が変更されたかを把握したうえで、必要に応じて簡単に以前のバージョンに戻せるように設計されたいくつかの機能があります。このセクションでは、Amazon Linux の 10 年以上の内部および外部使用 AWS から学んだ教訓について説明します。

⚠ Warning

`dnf --releasever=latest update` を実行することはベストプラクティスではありません。これを実行すると、OS の更新が最初に本番環境でテストされる結果を招く可能性があります。

`latest` を使用する代わりに、特定の AL2023 リリースバージョンを使用してください。これにより、以前にテストしたのと同じ変更を本稼働インスタンスにデプロイできます。たとえば、`dnf --releasever=2023.11.20260526 update` は常に 2023.11.20260526 リリースに更新されます。

詳細については、「[AL2023 ユーザーガイド](#)」の「[AL2023 の更新](#)」を参照してください。

OS 更新のデプロイの安全性を計画しない場合、アプリケーションやサービスと OS 更新の間で想定外のマイナスの相互作用が発生した際の影響が大幅に大きくなり、最悪の場合、全面的なサービス停止に至る可能性があります。ソフトウェアの問題と同様に、問題が検出されるのが早いほど、エンドユーザーへの影響が少なくなります。

OS 更新について、根本的に誤っている 2 つの思い込みに陥らないことが重要です。

1. OS ベンダーが OS の更新においてミスをするのではない。
2. ユーザーが依存する OS の特定の動作またはインターフェイスが、OS ベンダーが依存対象と見なす動作とインターフェイスと一致している。

つまり、OS ベンダーとユーザーが、更新に問題があったと同意できる。

善意に頼らず、あらゆる OS の更新において、デプロイの安全性を確保できるシステムを構築する必要があります。

本番環境にデプロイして、新しい OS 更新をテストすることはお勧めしません。ベストプラクティスは、OS をデプロイの構成要素の 1 つと見なし、本番環境への他の変更に適していると思われるものと同様のデプロイ安全メカニズムの適用を検討することです。

ベストプラクティスは、本番環境システムにデプロイする前に、すべての OS 更新をテストすることです。デプロイ時は、段階的なロールアウトと適切なモニタリングの併用をお勧めします。段階的なロールアウトにより、問題が発生した場合でも、即時ではないにしても影響をフリートのサブセットに限定できます。また、調査と影響の軽減を実施する間、その後の更新のデプロイを停止できます。

OS の更新による悪影響の軽減は、問題がどこにあっても、多くの場合において最優先事項であり、その次が問題の解決になります。OS 更新の導入が悪影響と関連する場合に、正常動作が確認されている以前の OS のバージョンにロールバックできる機能は強力なツールとなります。

Amazon Linux 2023 では、OS (または個々のパッケージ) のバージョンへの変更が繰り返し可能になる強力な新機能の [バージョン管理されたリポジトリを介した確定的なアップグレード](#) が導入されています。そのため、ある OS バージョンから次の OS バージョンへの移行時に問題が発生した場合でも、問題の解決策を検討しながら、既知の正常動作バージョンの使用を継続するため簡単なメカニズムが利用できます。

AL2023 では、新しいパッケージ更新をリリースするたびに、ロックする新しいバージョンと、そのバージョンにロックされる新しい AMI があります。[AL2023 リリースノート](#)では、各リリースの変更と、パッケージの更新で対処されたセキュリティ上の問題[AL2023 に関する Amazon Linux セキュリティアドバイザリ](#)について説明します。

例えば、[2023.6.20241028](#) リリースに存在する問題の影響を受けた場合、AMI とコンテナ イメージを使用して、以前の [2023.6.20241010](#) リリースに素早く戻すことができます。この場合、後続の [2023.6.20241031](#) リリースでパッケージのバグが修正されましたが、[バージョン管理されたリポジトリを介した確定的なアップグレード](#)により、影響を受けるすべてのユーザーが以前のイメージを使用するだけで、即座に即座に影響を緩和するための簡単な措置を取ることができました。

また、[バージョン管理されたリポジトリを介した確定的なアップグレード](#) は、進行中の OS 更新のデプロイが、現在の環境での更新または新たなAMIまたはコンテナイメージの起動によって、後続としてリリースされる OS 更新の影響を受けないことを保証します。

この最初の例では、フリート A は [2023.5.20241001](#) から [2023.6.20241010](#) リリースへの更新デプロイ途中の大規模なフリートです。ここで、[2023.6.20241028](#) リリースが公開されました。[バージョン管理されたリポジトリを介した確定的なアップグレード](#) は、フリート A のデプロイが適用される更新内容に変更なく継続されることを意味します。

最初にフリートの 1% にデプロイし、次に 5%、10%、20%、40% を 100% に達するまでデプロイするといった段階的なデプロイ戦略の目的は、より広範囲に展開する前に、限られた方法で変更をテストできるようにすることです。このタイプのデプロイ戦略は、本番環境の変更をデプロイするためのベストプラクティスであると一般的に考えられています。

段階的なデプロイ戦略とフリート A の [2023.6.20241010](#) への更新は、一度に多くのホストにデプロイされるステージにあり、[2023.6.20241028](#) がリリースされたという事実は、[バージョン管理されたリポジトリを介した確定的なアップグレード](#) を使用しているため、進行中のデプロイには影響しません。

フリート B が古いバージョンである [2023.5.20240708](#) を実行中で、[2023.6.20241028](#) への更新のデプロイを開始しており、フリート B がそのバージョンの問題の影響を受けた場合、デプロイの早い段階でこの問題に気付くでしょう。その時点で、当該問題の修正が利用可能になるまでロールアウトを一時停止するか、または同じバージョンである [2023.6.20241010](#) のデプロイを開始し、フリート B が [2023.5.20240708](#) ~ [2023.6.20241010](#) 間のすべての更新をフリート B が取得できるようにするかどうかについて、判断することができます。

OS の更新をすぐに適用しない問題が発生する可能性があることに注意してください。新しい更新には、環境に関連するバグやセキュリティの修正が含まれている可能性があります。詳細については、「[Amazon Linux 2023 でのセキュリティおよびコンプライアンス](#)」および「[AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)」を参照してください。

新しい OS 更新を簡単に取得し、本番環境へのデプロイ前にテストできるよう、デプロイシステムを適切に構成することが重要です。また、段階的なデプロイなどの仕組みを利用して、悪影響を最小限に抑えることも重要です。OS 更新による悪影響を軽減するために、デプロイ システムに正常動作が確認されている OS の以前のバージョンへ切り替える方法を知っておくことが重要です。問題の解決後は、当該旧バージョンの使用を継続するのではなく、新しく、正常動作が確認されているバージョンに移行します。

マイナーな更新のための準備

OS の小規模な更新 (AL2023 の新しいポイントリリースなど) に向けた準備は、ゼロに近い手間で済むように設計されています。今後の変更については、[AL2023 リリース ノート](#)を必ずお読みください。

[パッケージのサポート期間](#)が終了すると、言語ランタイムの新しいバージョン ([AL2023 での PHP](#) など) への移行が必要になる場合があります。ベストプラクティスは、サポート期間が終了する前に、新しい言語ランタイム バージョンに快適に移行することで、事前にこれに備えることです。

[pcre バージョン 1](#) などのパッケージの場合、事前に計画し、コードのいずれかを代替バージョンに移行する機会もあります。この場合、コードは pcre バージョン 2 です。万一の遅滞に備えて、できるだけ速やかに行うのがベストプラクティスです。

[Berkeley DB \(libdb\)](#) などに直接置き換えられない場合は、ユースケースに基づいた選択が必要になる場合があります。

メジャーな更新の準備

オペレーティング システムの新しいメジャー バージョンへの更新は、ほぼ普遍的に計画が必要とし、変更または廃止された機能に適応し、デプロイ前にテストを行う必要があります。Amazon

Linux 2023 の次のメジャー バージョンへの移行準備を段階的に進めることは珍しいことではありません。廃止された機能や削除された機能の使用に対応してから、次のメジャー バージョンへの移行を進めるといった方法をとることができます。

例えば、AL2 から AL2023 に移行する場合、[AL2 で廃止され、AL2023 で削除された機能](#) セクションを参照すると、AL2 を継続使用しながら AL2023 の準備をしている間に、安全で小さなステップでの段階的な移行が可能になります。例えば、yum パッケージ マネージャーなどの OS 使用以外のあらゆる [Python 2.7 は Python 3 に置き換えられました](#) の使用は、[AL2023 での Python](#) の使用に備えて Python 3 に移行できます。[PHP](#) を使用する場合、AL2 (PHP 8.2 [AL2 Extra](#) 経由) と AL2023 の両方が PHP 8.2 を搭載しているため、PHP バージョン移行と OS 移行の両方を同時に行う必要はありません。

AL2023 の使用中は、AL2023 を使用しながら、Amazon Linux 2023 の次のメジャー バージョンへの準備を今から行うこともできます。[AL2023 で廃止](#) セクションでは、AL2023 で廃止され、削除される予定の機能とパッケージについて説明します。

例えば、init スクリプトなどの残存する [System V init \(sysvinit\)](#) の使用を systemd に対応した同等のものに移行することで、将来へ備えるとともに、systemd 機能の全セットを使用して、サービスの監視、再起動の方法とその必要性、必要な他のサービス、リソースやアクセス許可の制限を適用する必要性の有無を判断できるようになります。

32 ビット サポートなどの機能の場合、廃止が OS の複数のメジャー バージョンにまたがる可能性があります。32 ビット版の場合、Amazon Linux 1 (AL1) は [32 ビット x86 \(i686\) AMI](#)、Amazon Linux 2 は [32 ビット x86 \(i686\) パッケージ](#)、Amazon Linux 2023 は [32bit x86 \(i686\) ランタイムのサポート](#) をそれぞれ廃止します。[IMDSv1](#) からの移行は、OS の複数のメジャー バージョンにも及びます。これらのタイプの変更については、一部のユーザーにおいて適応に長時間を要することが認識されているため、Amazon Linux 2023 で当該機能が利用できなくなるまで、十分な猶予期間が設定されています。

廃止機能のリストは OS のライフサイクルを通じて更新されるため、その最新の変更内容を常に把握されることをお勧めします。

新しいアップデートの通知を受け取る

新しい AL2023 AMI がリリースされると毎回通知を受け取ることができます。通知は、以下のトピックを使用して [Amazon SNS](#) で公開されます。

```
arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates
```

新しい AL2023 AMI が公開されると、メッセージがここに投稿されます。AMI のバージョンはメッセージに含まれます。

これらのメッセージは複数の方法で受信できます。次の方法を使用することをお勧めします。

1. [\[Amazon SNS コンソール\]](#) を開きます。
2. ナビゲーションバーで、必要に応じて、AWS リージョン を [米国東部 (バージニア北部)] に変更します。サブスクライブする SNS 通知が作成されたリージョンを選択する必要があります。
3. ナビゲーションペインで、[Subscriptions]、[Create subscription] の順に選択します。
4. [サブスクリプションの作成] ダイアログボックスで、次の操作を行います。
 - a. [トピック ARN] に、Amazon リソースネーム (ARN) **arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates** をコピーして貼り付けます。
 - b. [プロトコル] で [E メール] を選択します。
 - c. [エンドポイント] に、通知を受信するために使用できる E メールアドレスを入力します。
 - d. [サブスクリプションを作成] を選択します。
5. 「AWS 通知 - サブスクリプションの確定」という件名の確認メールを受け取ります。メールを開いて [サブスクリプションを確定] を選択して受信登録を完了します。

AL2023 でのバージョン管理されたりポジトリを介した確定的なアップグレード

Note

デフォルトでは、AL2023 インスタンスは起動時に追加のクリティカルかつ重要なセキュリティアップデートを自動的に受信しません。インスタンスには、AL2023 のバージョンで利用可能であった更新および選択した AMI が含まれています。

メジャーリリースとマイナーリリースから受け取る更新を管理する

AL2023 を使用すると、環境全体でパッケージバージョンと更新の一貫性を確保できます。同じ Amazon マシンイメージ (AMI) の複数のインスタンスで一貫性を保つこともできます。デフォルトで有効になっているバージョン化されたりポジトリによる確定的なアップグレード機能を使用すると、特定の必要に応じたスケジュールに基づいて更新を適用できます。

新しいパッケージ更新をリリースするたびに、ロックする新しいバージョンと、そのバージョンにロックされる新しい AMI があります。

AL2023 はリポジトリの特定のバージョンにロックされます。これはメジャーバージョンとマイナーバージョンの両方でサポートされています。SSM パラメータによって公開される AL2023 AMI は、常に最新バージョンです。重要かつ重要なセキュリティ更新を含む、最新のパッケージおよび更新が含まれています。

既存の AMI からインスタンスを起動した場合、更新は自動的に適用されません。プロビジョニングの一部としてインストールされたすべての追加パッケージは、既存の AMI のリポジトリバージョンにマップされます。

この機能では、環境全体でパッケージバージョンと更新の一貫性を確保する必要があります。これは、同じ AMI から複数のインスタンスを起動する場合に特に当てはまります。必要に応じたスケジュールに基づいて更新を適用できます。特定の更新セットを特定のリポジトリバージョンにロックすることもできるので、起動時に特定の更新セットを適用することもできます。

メジャーバージョンとマイナーバージョンのアップグレードの違い

AL2023 のメジャーバージョンリリースには大規模な更新が含まれており、パッケージを追加、削除、または更新する場合があります。互換性を確保するため、新しいメジャーバージョンでアプリケーションをテストしてからインスタンスを新しいメジャーバージョンにアップグレードします。

AL2023 のマイナーバージョンリリースには機能更新とセキュリティ更新が含まれていますが、パッケージの変更は含まれていません。これにより、Linux の機能とシステムライブラリ API を新しいバージョンでも引き続き利用できます。更新前にアプリケーションをテストする必要はありません。

更新が利用可能になる時期の確認

更新を適用するには、更新が利用可能であることを知り、更新のデプロイ方法を知っておく必要があります。

新しい AL2023 AMI のリリース時に派生した AMI を構築する場合は、[EC2 Image Builder](#) で AMI の自動的な構築、パッチ適用、テストを行うことができます。独自の AMI 構築パイプラインをトリガーしたり、ベース AMI を使用したりするには、[新しいアップデートの通知を受け取る](#) ことができます。

インプレースパッチ適用では、[AWS Systems Manager Patch Manager](#) などのツールを使用して、フリート全体への更新の適用をオーケストレーションできます。

AL2023 に基づく他のパブリック AMI の場合、AMI のプロバイダーごとにリリーススケジュールや通知方法は異なる場合があります。派生 AMI またはコンテナイメージを使用する場合は、更新のリリース時期についてパブリッシャーのドキュメントを確認してください。

各リリースの変更は、「[AL2023 リリースノート](#)」に記載されています。セキュリティアドバイザリは、[Amazon Linux セキュリティセンター \(ALAS\)](#) で公開されています。

AL2023 リポジトリから入手できるパッケージ更新を管理する

AL2023 リポジトリの新しいバージョンを公開しても、以前のバージョンはすべて引き続き利用できます。デフォルトでは、リポジトリのバージョンを管理するためのプラグインは、AMI の構築に使用されたのと同じバージョンにロックされます。パッケージの更新をコントロールする場合は、以下の手順に従います。

1. 以下のコマンドを実行して、使用可能なリポジトリバージョンを確認します。

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

2. 以下のコマンドを実行して、バージョンを選択します。

```
$ sudo dnf upgrade --releasever=version
```

このコマンドは、dnf を使用して現在の Amazon Linux リリースバージョンからコマンドラインで指定されたリリースバージョンへの更新を開始します。パッケージ更新のリストは dnf によって表示されます。更新が処理される前に、更新を確認する必要があります。更新が完了すると、新しいリリースバージョンが、今後のすべてのアクティビティに使用される dnf のデフォルトのリリースバージョンになります。

詳細については、「[AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)」を参照してください。

インスタンスの置き換えによる確定的な更新

「[AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)」で説明されている Amazon Linux 2023 の機能を使用すると、インスタンスの置き換えを簡単に行って、AL2023 の更新バージョンを確定的かつ安全にロールアウトできます。確定的な更新とは、新しいバージョンが段階的にロールアウトされる過程で、問題が見つかった場合に、問題の原因を特定しながら以前の AMI に簡単にロールバックできることを意味します。

インプレースでのパッチ適用ではなくインスタンスの置き換えを使用すると、新しいキャパシティの起動が、明確な状態 A と状態 B を持つ十分にテストされたコードパスとなり、更新がより確定的で予測可能になります。各前後の状態は、デプロイを開始する前に CI/CD システムで十分にテストできます。

インプレースでのパッチ適用を実行すると、更新プログラムを適用する前と後の間に多くの中間状態が生じ、状態のすべての組み合わせをテストするのが難しくなります。

インスタンスの置き換えを使用して確定的な更新を適用する OS 更新戦略は、blue/green、ウェーブ、フェーズベースのデプロイモデルに適しています。

バージョン管理されたリポジトリによる確定的アップグレードの使用

トピック

- [確定的なアップグレードシステムを使用する](#)
- [確定的なアップグレードされたシステムの選択的更新](#)
- [確定的なアップグレードでの永続的上書きの使用](#)

確定的なアップグレードシステムを使用する

Note

パッケージマネージャーのデフォルトの動作が AL2 から変更されました。

確定的アップグレードは、すべての変更を本番環境に本格的にデプロイする前に詳細にテストできる強力な方法です。新しい各 AL2023 AMI は、特定バージョンの AL2023 にロックされます。これにより、特定の AMI の起動時に、どのバージョンの OS パッケージをインストールするかが確定的に判断されます。インプレース更新はリリースバージョンを特定できるため、フリート全体の確定的な動作が保証されます。新しい AMI またはインプレース更新バージョンに移行する際は、各バージョンを CI/CD パイプラインでテストし、本番環境にデプロイする前に潜在的な問題を検出できます。

[AWS Systems Manager Patch Manager](#) などのツールを使用して、フリート全体での更新の適用をオーケストレーションできます。新しい AL2023 AMI のリリース時に派生 AMI を構築する場合は、[EC2 Image Builder](#) により、AMI の自動的な構築、パッチ適用、テストができます。または[新しいアップグレードの通知を受け取る](#)により、新しいベース AMI がいつ利用可能になるかを確認したり、独自の AMI 構築パイプラインをトリガーしたりできます。

特定のアドバイザリからの更新のみに制限する方法については、「[セキュリティ更新プログラムのインプレースでの適用](#)」を参照してください。

インプレースのパッチ適用には、dnf パッケージマネージャーを使用できます。dnf upgrade コマンドを実行すると、システムは releasever 変数が指定するリポジトリ内のアップグレードをチェックします。有効な releasever は、###バージョンまたは **2023.11.20260526** などの日付スタンプ付きバージョンです。

以下のいずれかの方法を使用して、releasever の値を変更できます。これらの方法は、システム優先度の降順で一覧表示されています。つまり、方法 1 は方法 2 と 3 を上書きし、方法 2 は方法 3 を上書きします。

1. コマンドラインフラグ内の値 `--releasever=latest` (使用する場合)。
2. 上書き変数ファイルで指定されている値 `/etc/dnf/vars/releasever` (設定されている場合)。
3. 現在インストールされている system-release パッケージのバージョン。

以下の例ではバージョンは、**2023.0.20230210** です。

```
$ rpm -q system-release
system-release-2023.0.20230210-0.amzn2023.noarch
```

新しくインストールされたシステムには、上書き変数は存在しません。システムはインストールした system-release のバージョンのみにロックされているため、アップグレードはできません。

```
$ 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!
```

releasever フラグを使用して必要なバージョンを指定することで、特定のバージョンのパッケージを取得できます。

```
$ 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
```

--releasever オプションは system-release と /etc/dnf/vars/releasever の両方を上書きするため、このアップグレードの結果は以下のようになります。

1. アップグレードにより、以前のバージョンと新しいバージョンとの間で変更されたインストール済みパッケージがすべて置き換えられます。
2. アップグレードすると、システムは新しい system-release のバージョンのリポジトリにロックされます。

どの releasever (AL2023 リリース) を更新するかを常に指定することで、フリート全体での変更を確定的に把握できます。バージョン **A** を起動して、**B** に更新し、さらに **C** に更新しました。

確定的なアップグレードされたシステムの選択的更新

Note

特定の更新を選択するのではなく、新しいリリースのすべての更新をインストールすることをお勧めします。OS に更新の一部のみを適用することは、更新全体を適用するという標準プラクティスの例外です。

システムを元のリリースバージョンにロックしたまま、最近のリリースから選択したパッケージをインストールするとします。

`dnf check-update` を使用して、アップグレードするパッケージを特定します。

```
$ 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

アップグレードしたいパッケージをインストールします。sudo dnf upgrade --releasever=latest とパッケージ名を使用して、system-release パッケージが変更されないようにします。

```
$ 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

sudo dnf upgrade --releasever=latest を使用すると、system-release を含むすべてのパッケージが更新されます。その後、永続的な上書きを設定しない限り、バージョンは新しい system-release にロックされたままになります。

確定的なアップグレードでの永続的上書きの使用

Note

確定的な更新により、OS の変更を CI/CD パイプラインに統合できます。確定的な更新を無効にすると、デプロイ前にテストできなくなります。

`--releasever=latest` を追加する代わりに、変数値を `###` 設定することで、永続的な上書きを使用してシステムのロックを解除できます。latest を常に使用すると、AL2023 の動作は AL2 更新モデルに戻ります。パッケージマネージャーへのすべての呼び出しは、常に最新のリリースを参照し、特定のバージョンの OS にロックされません。

Warning

確定的な更新の永続的な上書きを使用してパッケージマネージャーをロック解除することで、アプリケーションと本番環境の OS 更新との潜在的な非互換性が判明するというリスクがあります。

非互換性はまれに存在しますが、OS 更新で新しいコード変更を環境に統合する場合、統合テストにより、本番環境に悪影響を及ぼすコード変更のデプロイを防止できます。

```
$ 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

上書き変数 `/etc/dnf/vars/releasever` を使用した場合は、以下のコマンドを使用して上書き値を消去してデフォルトのロック動作に戻します。

```
$ sudo rm /etc/dnf/vars/releasever
```

特定のバージョンではなく、`latest` の使用に対する永続的な上書きの使用は、AL2 のデフォルトの動作に似ています。AL2 に基づく AMI を構築するサービスの中には、この動作を無効にして、AL2023 でデフォルトで取得される特定のパッケージバージョンにロックするものがあります。

確定的な更新を無効にするのではなく、インスタンスを新しい AMI から起動したインスタンスに置き換えることをお勧めします。インスタンスの置き換えが不可能な場合は、[AWS Systems Manager Patch Manager](#) などのツールを使用して、フリート全体への更新の適用をオーケストレーションすることをお勧めします。[EC2 Image Builder](#) により、AL2023 ベースイメージから派生した独自の AMI の自動的な構築、パッチ適用、テストを行うこともできます。また、[新しいアップデートの通知を受け取る](#) ことで、独自の AMI 構築パイプラインをトリガーすることもできます。

本番稼働前の環境で `latest` を使用し、次に `latest` を使用して本番稼働環境にデプロイすることは、OS 更新とアプリケーション間の問題から保護することになりません。新しい AL2023 リリースは随時発生する可能性があるため、本番環境での `latest` のすべての使用にはリスクが伴います。

AL2023 のパッケージとオペレーティングシステムとの更新を管理する

Amazon Linux の以前のバージョンとは異なり、AL2023 AMI の Amazon Linux リポジトリは特定のバージョンにロックされています。AL2023 インスタンスにセキュリティ修正とバグ修正の両方を適用するには、DNF の設定を最新の利用可能なリリースバージョンに更新します。または、新しい AL2023 インスタンスを起動します。

このセクションでは、実行中のインスタンスの DNF パッケージとリポジトリを管理する方法について説明します。また、ユーザーデータスクリプトから、使用可能な最新の Amazon Linux リポジトリを起動時に有効にするように DNF を設定する方法についても説明します。詳細については、「[DNF コマンドリファレンス](#)」を参照してください。

新しい AL2023 リリースで利用可能なすべての更新を適用することをお勧めします。セキュリティ更新プログラムのみ、または特定の更新プログラムのみを選択するのは例外とし、ルールにはしないでください。[セキュリティアドバイザリ](#) が特定のインスタンスに関連するリストについては、「[適用可能なアドバイザリの一覧表示](#)」を参照してください。特定の[アドバイザリ](#)に関連する更新のみインストールする方法については、「[セキュリティ更新プログラムのインプレースでの適用](#)」を参照してください。

Important

脆弱性を報告する場合、または AWS クラウドサービスやオープンソースプロジェクトに関するセキュリティ上の懸念がある場合は、「[脆弱性レポート](#)」ページを使用して AWS セキュリティにお問い合わせください。

トピック

- [使用可能なパッケージ更新の確認](#)
- [DNF およびリポジトリバージョンを使用してセキュリティ更新を適用します。](#)
- [\(セキュリティ\) 更新後の自動サービス再起動](#)
- [セキュリティ更新を適用する際、いつ再起動が必要になりますか？](#)
- [最新のリポジトリバージョンを有効にしたインスタンスの起動](#)
- [パッケージサポート情報の取得](#)
- [dnf check-release-update を使用した新しいリポジトリバージョンの確認](#)
- [新しいリポジトリの追加、有効化、無効化](#)

- [cloud-init によるリポジトリの追加](#)

使用可能なパッケージ更新の確認

`dnf check-update` コマンドを使用して、システムの更新を確認できます。AL2023 の場合は、コマンドに `--releasever=version-number` オプションを追加することをお勧めします。

このオプションを追加すると、DNF は新しいバージョンのリポジトリの更新も確認します。例えば、`dnf check-update` コマンドを実行した後は、返された最新のバージョンを `version-number` の値として使用します。

最新バージョンのリポジトリを使用するようにインスタンスを更新すると、出力には更新対象となるすべてのパッケージの一覧が含まれます。

Note

`dnf check-update` コマンドにオプションフラグを付けてリリースバージョンを指定しない場合、現在設定されているリポジトリバージョンのみが確認されます。つまり、新しいバージョンのリポジトリにあるパッケージは確認されません。

Updates in a specific version

この例では、[2023.0.20230315](#) リリースのコンテナを起動した場合、どの更新が [2023.1.20230628](#) リリースで利用可能であるかを確認します。

Note

この例では、[2023.0.20230315](#) および [2023.1.20230628](#) リリースを使用しています。これらは AL2023 の最新リリースではありません。最新のセキュリティ更新プログラムを含む最新リリースについては、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.0.20230315](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の `.0` は、特定のリリースにおけるイメージのバージョンを示します。このイメージのバージョンは通常 0 です。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこから更新を確認します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

`dnf check-update` コマンドは、[2023.1.20230628](#) リリースで利用可能な更新を確認するために使用されます。

Note

パッケージの更新の適用は特権オペレーションです。コンテナで実行する場合は、通常、権限昇格は必要ありませんが、Amazon EC2 インスタンスなどのコンテナ化されていない環境で実行する場合は、権限を昇格せずに更新を確認できます。

```
$ 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 パッケージのバージョンには、dnf upgrade コマンドが更新するリリースが表示されます。これは、dnf check-update --releasever=**2023.1.20230628** コマンドで指定された [2023.1.20230628](#) リリースです。

Updates in the latest version

この例では、[2023.4.20240319](#) リリースのコンテナを起動した場合、どの更新が AL2023 の latest バージョンで利用可能であるかを確認します。執筆時点で、latest リリースは [2023.5.20240708](#) であるため、この例に記載されている更新は、そのリリース時点のものです。

Note

この例では [2023.4.20240319](#) リリースと [2023.5.20240708](#) リリースを使用しており、後者は執筆時点での最新リリースです。最新リリースの詳細については、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.4.20240319](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の .1 は、特定のリリースにおけるイメージのバージョンを示します。イメージバージョンは通常ゼロですが、この例ではイメージバージョンが 1 のリリースを使用します。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこから更新を確認します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

dnf check-update コマンドは、執筆時点で [2023.5.20240708](#) であった latest リリースで利用可能な更新を確認するために使用されます。

Note

パッケージの更新の適用は特権オペレーションです。コンテナで実行する場合は、通常、権限昇格は必要ありませんが、Amazon EC2 インスタンスなどのコンテナ化されていない環境で実行する場合は、権限を昇格せずに更新を確認できます。

```
$ 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
bash-5.2#
```

4.14.0-1.amzn2023.0.5

amazonlinux

system-release パッケージのバージョンは、dnf upgrade コマンドによって更新されるリリースを示します。

このコマンドでは、新しいパッケージが使用可能な場合、返されるコードは 100 です。新しいパッケージがない場合、返されるコードは 0 です。さらに、出力には更新予定のパッケージもすべて一覧表示されます。

DNF およびリポジトリバージョンを使用してセキュリティ更新を適用します。

新しいパッケージ更新およびセキュリティ更新は、新しいリポジトリバージョンでのみ使用可能です。以前の AL2023 AMI バージョンから起動したインスタンスの場合、セキュリティ更新をインストールする前にリポジトリバージョンを更新する必要があります。dnf check-release-update コマンドには、システムにインストールされているすべてのパッケージを新しいリポジトリのバージョンに更新する更新コマンドの例が含まれています。

Note

dnf check-update コマンドにオプションフラグを付けてリリースバージョンを指定しない場合、現在設定されているリポジトリバージョンのみが確認されます。つまり、リポジトリの後続バージョンに存在するインストール済みパッケージの更新は適用されません。

このセクションでは、個々の更新またはセキュリティ更新としてマークされた更新のみを選択および選択するのではなく、利用可能なすべての更新を適用する推奨アップグレードパスについて説明します。すべての更新を適用することで、既存のインスタンスは、更新された AMI の起動と同じパッケージセットに移行します。この一貫性により、フリート全体のパッケージバージョンのばらつきが軽減されます。特定の更新の適用の詳細については、「[セキュリティ更新プログラムのインプレースでの適用](#)」を参照してください。

Applying updates in a specific version

この例では、[2023.1.20230628](#) リリースのコンテナを起動した場合、[2023.0.20230315](#) リリースで利用可能な更新を適用します。

Note

この例では、[2023.0.20230315](#) および [2023.1.20230628](#) リリースを使用しています。これらは AL2023 の最新リリースではありません。最新のセキュリティ更新プログラムを含む最新リリースについては、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.0.20230315](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の `.0` は、特定のリリースにおけるイメージのバージョンを示します。このイメージのバージョンは通常 `0` です。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこから更新を適用します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

`dnf upgrade` コマンドを使用して、[2023.1.20230628](#) リリースに存在するすべての更新を適用できるようになりました。

Note

パッケージの更新の適用は特権オペレーションです。コンテナで実行する場合、通常、権限昇格は必要ありませんが、Amazon EC2 インスタンスなどのコンテナ化されていない環境で実行する場合は、`root` ユーザーとして `dnf upgrade` コマンドを実行する必要があります。これは、`sudo` または `su` コマンドを使用して行うことができます。

```
$ 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 パッケージのバージョンには、dnf upgrade コマンドが更新するリリースが表示されます。これは、dnf upgrade --releasever=**2023.1.20230628** コマンドで指定された [2023.1.20230628](#) リリースです。

デフォルトでは、dnf は更新を適用することを確認するよう求めます。このプロンプトは、-y フラグを使用して dnf にバイパスできます。この例では、dnf upgrade -y --

releasever=**2023.1.20230628** コマンドは更新を適用する前に確認を求めません。これは、スクリプトやその他のオートメーション環境で役立ちます。

更新の適用が確認されると、dnf は更新を適用します。

```
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

この例では、[2023.4.20240319](#) リリースのコンテナを起動した場合に、AL2023 の latest バージョンで利用可能な更新を適用します。執筆時点で、latest リリースは [2023.5.20240708](#) であるため、この例に記載されている更新は、そのリリース時点のものです。

Note

この例では [2023.4.20240319](#) リリースと [2023.5.20240708](#) リリースを使用しており、後者は執筆時点での最新リリースです。最新リリースの詳細については、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.4.20240319](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の .1 は、特定のリリースにおけるイメージのバージョンを示します。イメージバージョンは通常ゼロですが、この例ではイメージバージョンが 1 のリリースを使用します。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこから更新を適用します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1  
bash-5.2#
```

dnf upgrade コマンドは、執筆時点で [2023.5.20240708](#) であった latest リリースで利用可能な更新を適用するために使用されます。

Note

パッケージの更新の適用は特権オペレーションです。コンテナで実行する場合、通常、権限昇格は必要ありませんが、Amazon EC2 インスタンスなどのコンテナ化されていない環境で実行する場合は、root ユーザーとして `dnf upgrade` コマンドを実行する必要があります。これは、`sudo` または `su` コマンドを使用して行うことができます。

デフォルトでは、`dnf` は更新を適用することを確認するよう求めます。この例では、`dnf` への `-y` フラグを使用して、このプロンプトをバイパスします。

```
$ 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#

AL2023 の更新を確認するには、以下のうち 1 つ以上を実行します。

- `dnf check-update` コマンドを実行します。これにより、ロックされている Amazon Linux のバージョンに未適用の更新があるかどうかを確認します。これは、`system-release` パッケージのみを更新した場合に更新が表示されることがあります。この操作は、インスタンスがロックされるリポジトリのバージョンを切り替えるだけで、そのリポジトリに含まれる更新は適用されません。
- Amazon Linux リポジトリの更新 SNS トピック (`arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates`) をサブスクライブします。詳細については、「[Amazon Simple 通知サービス デベロッパーガイド](#)」の「Amazon SNS トピックのサブスクライブ」を参照してください。
- 「[AL2023 リリースノート](#)」を定期的に参照してください。
- [dnf check-release-update を使用した新しいリポジトリバージョンの確認](#) で新しいバージョンを見つけます。

⚠ Important

セキュリティ更新プログラムを含む AL2023 の新しいバージョンは頻繁にリリースされます。関連するセキュリティパッチを常に最新の状態に保つようにしてください。

(セキュリティ) 更新後の自動サービス再起動

Amazon Linux に [smart-restart](#) パッケージが付属するようになりました。Smart-restart は、システムのパッケージマネージャーを使用してパッケージがインストールまたは削除される際に、システム更新に伴って systemd サービスを再起動します。これは、dnf (update|upgrade|downgrade) が実行されるたびに行われます。

Smart-restart は、dnf-utils からの needs-restarting パッケージとカスタム拒否リストのしくみを使用して、どのサービスを再起動する必要があるか、およびシステムの再起動が推奨されるかどうかを判断します。システムの再起動が推奨されている場合、再起動ヒントマーカーファイルが生成されます (/run/smart-restart/reboot-hint-marker)。

smart-restart をインストールするには

次の DNF コマンドを実行します (他のパッケージと同様に)。

```
$ sudo dnf install smart-restart
```

インストール後、後続のトランザクションによって smart-restart ロジックがトリガーされます。

拒否リスト

Smart-restart は、特定のサービスの再起動をブロックするように指示することができます。ブロックされたサービスは、再起動が必要かどうかの判断には影響しません。追加のサービスをブロックするには、次の例に示すように、/etc/smart-restart-conf.d/ にサフィックス -denylist が付いたファイルを追加します。

```
$ cat /etc/smart-restart-conf.d/custom-denylist
# Some comments
myservice.service
```

Note

再起動が必要かどうかを決定するときに、すべての `*-denylist` ファイルが読み取られ、評価されます。

カスタムフック

拒否リストに加えて、`smart-restart` はサービスを再起動する前後にカスタムスクリプトを実行するしくみを提供します。カスタムスクリプトを使用して、手動で準備ステップを実行したり、他のコンポーネントに再起動が残っていることや完了したことを通知したりするために使用できます。

`/etc/smart-restart-conf.d/` 内にある、サフィックスが `-pre-restart` または `-post-restart` のすべてのスクリプトが実行されます。順序が重要な場合は、次の例に示すように、すべてのスクリプトに数字を付け、実行順序を確認します。

```
$ ls /etc/smart-restart-conf.d/*-pre-restart
001-my-script-pre-restart
002-some-other-script-pre-restart
```

セキュリティ更新を適用する際、いつ再起動が必要になりますか？

状況によっては、更新を適用するために Amazon Linux を再起動する必要があります。

- Linux カーネルパッケージの更新には、最新のセキュリティ更新を使用して新しいカーネルをアクティブ化するための再起動が必要です。カーネルライブパッチにより、セキュリティ更新を一定期間延期できる場合があります。詳細については、「[AL2023 のカーネルライブパッチ](#)」を参照してください。
- EC2 Metal インスタンスでは、Amazon Linux はマイクロコードの更新を提供します (Intel CPUs 用に `microcode_ctl` パッケージ、AMD CPU 用に `amd-ucode-firmware` パッケージを使用)。これらのマイクロコード更新は、以降のインスタンスの再起動時にのみアクティブ化されます。仮想化された EC2 インスタンスの場合、基盤となる [AWS Nitro システム](#)がマイクロコードの更新を処理します。
- 実行中の `systemd` サービスの中には、システムが完全に再起動した後にのみ正しく機能するものがあります。この `smart-restart` のしくみは、再起動のヒントを残すことで、このような状況を通知します。「[\(セキュリティ\) 更新後の自動サービス再起動](#)」を参照してください。

最新のリポジトリバージョンを有効にしたインスタンスの起動

ユーザーデータスクリプトに DNF コマンドを追加して、Amazon Linux AMI の起動時にどの RPM パッケージをインストールするかを制御できます。以下の例では、ユーザーデータスクリプトを使用して、ユーザーデータスクリプトを使用して起動されるすべてのインスタンスに同じパッケージ更新がインストールされていることを確認します。

```
#!/bin/bash
dnf upgrade --releasever=2023.0.20230210
# Additional setup and install commands below
dnf install httpd php7.4 mysql80
```

このスクリプトは superuser (root) として実行する必要があります。これを行うには、以下のコマンドを実行します。

```
$ sudo sh -c "bash nameofscript.sh"
```

詳細については、「Amazon EC2 ユーザーガイド」の [「ユーザーデータおよびシェルスクリプト」](#) を参照してください。

Note

ユーザーデータスクリプトを使用する代わりに、最新の Amazon Linux AMI を起動するか、Amazon Linux AMI をベースにしたカスタム AMI を起動します。最新の Amazon Linux AMI には必要な更新がすべてインストールされており、特定のリポジトリバージョンを指すように設定されています。

パッケージサポート情報の取得

AL2023 には、さまざまなオープンソースソフトウェアプロジェクトが組み込まれています。これらのプロジェクトはそれぞれ Amazon Linux とは独立して管理されており、リリースやサポート終了のスケジュールも異なります。これらのさまざまなパッケージに関する Amazon Linux 固有の情報を提供するために、DNF supportinfo プラグインはパッケージに関するメタデータを提供します。以下の例では、**dnf supportinfo** コマンドは glibc パッケージのメタデータを返します。

```
$ 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.
```

パッケージのサポート情報は、「[AL2023 リリースノート](#)」の「[サポートステートメント](#)」セクションでも入手できます。

dnf check-release-update を使用した新しいリポジトリバージョンの確認

AL2023 インスタンスでは、DNF ユーティリティを使用してリポジトリを管理し、更新された RPM パッケージを適用できます。これらのパッケージは Amazon Linux リポジトリにあります。DNF コマンド `dnf check-release-update` を使用して、DNF リポジトリの新しいバージョンを確認できます。

Note

AL2023 コンテナイメージには、デフォルトで `dnf check-release-update` コマンドは含まれません。

```
$ 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)'"
```

`dnf install 'dnf-command(check-release-update)'` が実行されると、dnf は `check-release-update` コマンドを提供する `dnf-plugin-release-notification` パッケージをインストールします。次の例では、dnf に `-q` 引数を渡すことで出力を抑制します。

```
$ dnf -y -q install 'dnf-command(check-release-update)'
Installed:
```

```
dnf-plugin-release-notification-1.2-1.amzn2023.0.2.noarch
```

Amazon EC2 インスタンスなどのコンテナ化されていない環境では、`check-release-update` コマンドはデフォルトで含まれます。

```
$ 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
```

これにより、使用可能なすべての新しいバージョンの DNF リポジトリのすべてのリストが返されます。何も返されない場合は、DNF が現在使用可能な最新バージョンを使用するように設定されていることを意味します。現在インストールされている `system-release` パッケージのバージョンによって `releasever` DNF 変数が設定されます。現在のリポジトリバージョンを確認するには、以下のコマンドを実行します。

```
$ rpm -q system-release --qf "%{VERSION}\n"
```

DNF パッケージトランザクション (インストール、更新、削除コマンドなど) を実行すると、リポジトリの新しいバージョンを知らせる警告メッセージが表示されます。例えば、AL2023 の古いバージョンから起動されたインスタンスに `httpd` パッケージをインストールすると、以下の出力が返されます。

```
$ 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!

新しいリポジトリの追加、有効化、無効化

Warning

AL2023 で使用するように設計されたリポジトリのみを追加します。
他のディストリビューション向けに設計されたリポジトリは、現時点では動作する場合がありますが、AL2023 のパッケージ更新や、AL2023 向けに設計されていないリポジトリの更新によって、今後も動作し続ける保証はありません。

デフォルトの Amazon Linux リポジトリとは異なるリポジトリからパッケージをインストールするには、リポジトリの場所がわかるように DNF パッケージ管理システムを設定する必要があります。

dnf にパッケージリポジトリを認識させるには、そのリポジトリの情報を `/etc/yum.repos.d/` ディレクトリ内の設定ファイルに追加します。多くのサードパーティーリポジトリは、設定ファイルの内容または設定ファイルを含むインストール可能なパッケージを提供します。

Note

リポジトリは `/etc/dnf/dnf.conf` ファイルで直接設定できますが、これはお勧めしません。各リポジトリは、`/etc/yum.repos.d/` の独自のファイルで設定することをお勧めします。

現在有効になっているリポジトリを確認するには、以下のコマンドを実行します。

```
$ 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

--verbose オプションフラグを追加しない場合、出力には、Repo-id、Repo-name、および Repo-status の情報のみが含まれます。

yum リポジトリを /etc/yum.repos.d ディレクトリに追加する方法

1. .repo ファイルの場所を検索します。この例では、.repo ファイルは、<https://www.example.com/repository.repo> にあります。
2. dnf config-manager コマンドを使用してリポジトリを追加します。

```
$ 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
```

リポジトリをインストールしたら、以下の手順で説明するように有効にする必要があります。

/etc/yum.repos.d で yum リポジトリを有効にするには、--enable フラグと **#####** 名を指定して dnf config-manager コマンドを実行します。

```
$ sudo dnf config-manager --enable repository
```

 Note

リポジトリを無効にするには、同じコマンド構文を使用しますが、コマンド内の --enable を --disable に置き換えます。

cloud-init によるリポジトリの追加

前の方法でリポジトリを追加するほかに、cloud-init フレームワークを使用して新しいリポジトリを追加することもできます。

新しいパッケージリポジトリを追加するには、以下のテンプレートを使用することをお勧めします。このファイルをローカルに保存することを検討してください。

```
#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

cloud-init を使用する利点の 1 つは、設定ファイルに `packages:` セクションを追加できることです。このセクションには、インストールするパッケージの名前を含めることができます。パッケージは、デフォルトのリポジトリからも、cloud-config ファイルに追加した新しいリポジトリからもインストールできます。

YAML ファイルの構造に関するより具体的な情報については、「*cloud-init* ドキュメント」の「[YUM リポジトリの追加](#)」を参照してください。

YAML フォーマットファイルを設定したら、AWS CLIの cloud-init フレームワークで実行できます。必要な操作を呼び出すための `--userdata` オプションと `.yaml` ファイル名を必ず含めてください。

```
$ 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.yaml
```

AL2023 のカーネルライブパッチ

AL2023 のカーネルライブパッチにより、実行中のアプリケーションを再起動や中断せずに、実行中の Linux カーネルに特定のセキュリティの脆弱性や重大なバグのパッチを適用することができます。また、カーネルライブパッチを利用することで、システムを再起動できるまで修正を適用しながらアプリケーションの可用性を向上させることができます。

AWS は、AL2023 用の 2 種類のカーネルライブパッチをリリースします。

- **セキュリティ更新** – Linux の共通脆弱性とエクスポージャー (CVE) の更新プログラムが含まれます。これらの更新プログラムは、通常、Amazon Linux Security Advisory の評価で Important または Critical と評価されます。場合によっては、CVE が割り当てられる前に更新を提供する AWS ことがあります。そのような場合、パッチはバグ修正プログラムとして提供される場合があります。セキュリティ更新用のカーネルライブパッチはベストエフォートベースで提供されます。すべての重要または重要な CVEs が、特定のカーネルバージョンのライブパッチとして利用できるわけではありません。特定の CVE がライブパッチで対処されているかどうかを判断するには、[Amazon Linux Security Advisories](#) を確認してください。
- **バグ修正** – CVE に関連付けられていない重大なバグや安定性の問題の修正プログラムが含まれます。

AWS は、AL2023 カーネルバージョンのカーネルライブパッチをリリース後最大 3 か月間提供します。その後、カーネルライブパッチを引き続き入手するには、新しいカーネルバージョンに更新する必要があります。

AL2023 のカーネルライブパッチは、既存の AL2023 リポジトリから署名付きの RPM パッケージとして入手できます。パッチを個別のインスタンスにインストールするには、既存の DNF パッケージマネージャーワークフローを使用できます。または、AWS Systems Manager を使用してマネージドインスタンスのグループにインストールすることもできます。

AL2023 のカーネルライブパッチは、追加料金なしで提供されます。

トピック

- [制限事項](#)
- [サポートされている構成と前提条件](#)
- [カーネルライブパッチを使用する](#)

制限事項

カーネルライブパッチの適用中は、休止を実行したり、高度なデバッグツール (SystemTap、kprobes、eBPF ベースのツールなど) を使用したり、カーネルライブパッチを適用したインフラストラクチャで使用されている ftrace の出力ファイルにアクセスしたりすることはできません。

Note

技術的な制限により、ライブパッチの適用では一部の問題に対処できません。このため、これらの修正はカーネルライブパッチパッケージに含まれず、ネイティブカーネルパッケージの更新でのみ提供されます。ネイティブカーネルパッケージをインストールし、システムを[更新および再起動](#)して通常どおりパッチをアクティブ化できます。

サポートされている構成と前提条件

カーネルライブパッチは、Amazon EC2 インスタンスおよび AL2023 が実行されているオンプレミスの仮想化マシンでサポートされています。

AL2023 でカーネルライブパッチを使用するには、以下を使用する必要があります。

- 64 ビット x86_64 または ARM64 アーキテクチャ
- カーネルバージョン 6.1 または 6.12

ポリシーの要件

AL2023 リポジトリからパッケージをダウンロードするには、Amazon EC2 からサービス所有の Amazon S3 バケットにアクセスする必要があります。環境で Amazon S3 用に Amazon Virtual Private Cloud (VPC) エンドポイントを使用している場合、VPC エンドポイントポリシーがそれらのパブリックバケットへのアクセスを許可する必要があります。次の表は、Amazon EC2 が Kernel Live Patching のためにアクセスが必要となる可能性がある Amazon S3 バケットを示しています。

S3 バケット ARN	説明
arn:aws:s3:::al2023-repos- <i>region</i> -de612dc2/*	AL2023 リポジトリを含む Amazon S3 バケット

カーネルライブパッチを使用する

カーネルライブパッチを有効にして個別のインスタンスで使用するには、インスタンス自体でコマンドラインを使用できます。Systems Manager を使用して、カーネルライブパッチを有効にしてマネージドインスタンスのグループで使用するには、AWS Systems Manager を使用できます。

以下のセクションでは、コマンドラインを使用して、カーネルライブパッチを有効にして個別のインスタンスで使用方法について説明します。

マネージドインスタンスのグループでカーネルライブパッチを有効にして使用方法の詳細については、「AWS Systems Manager ユーザーガイド」の「[AL2023 インスタンスでカーネルライブパッチを使用する](#)」を参照してください。

トピック

- [カーネルライブパッチの有効化](#)
- [利用可能なカーネルライブパッチを表示する](#)
- [カーネルライブパッチを適用する](#)
- [適用されたカーネルライブパッチの表示](#)
- [カーネルライブパッチの無効化](#)

カーネルライブパッチの有効化

AL2023 では、カーネルライブパッチはデフォルトでは無効になっています。ライブパッチを使用するには、カーネルライブパッチの DNF プラグインをインストールして、ライブパッチ機能を有効にする必要があります。

カーネルライブパッチを有効にする方法

1. カーネルライブパッチは、カーネルバージョン 6.1 以降の AL2023 で使用できます。カーネルバージョンを確認するには、次のコマンドを実行します。

```
$ sudo dnf list kernel
```

2. カーネルライブパッチの DNF プラグインをインストールします。

```
$ sudo dnf install -y kpatch-dnf
```

3. カーネルライブパッチの DNF プラグインを有効にします。

```
$ sudo dnf kernel-livepatch -y auto
```

このコマンドは、設定されているリポジトリから最新バージョンのカーネルライブパッチの RPM もインストールします。

4. カーネルライブパッチの DNF プラグインが正常にインストールされたことを確認するには、以下のコマンドを実行します。

カーネルライブパッチを有効にすると、空のカーネルライブパッチの RPM が自動的に適用されます。カーネルライブパッチが正常に有効になっていれば、このコマンドは最初の空のカーネルライブパッチの RPM (およびライブパッチを含む DNF リポジトリを設定するもうひとつの RPM)を含むリストを返します。

```
$ 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. kpatch パッケージをインストールします。

```
$ sudo dnf install -y kpatch-runtime
```

6. 既にインストール済みの場合は、kpatch サービスを更新します。

```
$ sudo dnf upgrade kpatch-runtime
```

7. kpatch サービスを起動します。このサービスは、初期化時または起動時にすべてのカーネルライブパッチをロードします。

```
$ sudo systemctl enable kpatch.service && sudo systemctl start kpatch.service
```

利用可能なカーネルライブパッチを表示する

Amazon Linux のセキュリティアラートは、Amazon Linux Security Center に公開されます。カーネルライブパッチのアラートを含む AL2023 のセキュリティアラートの詳細については、「[Amazon Linux Security Center](#)」を参照してください。カーネルライブパッチには、ALASLIVEPATCH というプレフィクスが付きます。Amazon Linux Security Center では、バグに対応するカーネルライブパッチは一覧に表示されていない場合があります。

アドバイザリおよび CVE に対する利用可能なカーネルライブパッチは、コマンドラインを使用して見つけることもできます。

アドバイザリに対する利用可能なすべてのカーネルライブパッチを一覧表示するには次のコマンドを使用します。

```
$ 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
```

CVE に対する利用可能なすべてのカーネルライブパッチを一覧表示するには次のコマンドを使用します。

```
$ 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
```

カーネルライブパッチを適用する

カーネルライブパッチは、DNF パッケージマネージャーを使用して、通常の更新プログラムを適用するのと同じ方法で適用します。カーネルライブパッチの DNF プラグインは、適用可能なカーネルライブパッチを管理します。

Tip

カーネルライブパッチを使用して定期的にカーネルを更新し、システムが再起動されるまで特定のセキュリティ修正を受け取ることをお勧めします。ライブパッチとしてデプロイ追加の修正がネイティブのカーネルパッケージに含まれているかどうかを確認し、その場合はカーネルを[更新および再起動](#)してください。

特定のカーネルライブパッチを適用するか、利用可能なカーネルライブパッチを定期的なセキュリティ更新プログラムと一緒に適用するかを選択できます。

特定のカーネルライブパッチを適用するには

1. 「[利用可能なカーネルライブパッチを表示する](#)」で説明されているコマンドのいずれかを使用して、カーネルライブパッチのバージョンを取得します。
2. AL2023 カーネルのカーネルライブパッチを適用します。

```
$ sudo dnf install kernel-livepatch-kernel_version-package_version.amzn2023.x86_64
```

例えば、以下のコマンドは、AL2023 カーネルバージョン 6.1.12-17.42 のカーネルライブパッチを適用します

```
$ sudo dnf install kernel-livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
```

利用可能なカーネルライブパッチを定期的なセキュリティ更新プログラムと一緒に適用する方法
次のコマンドを使用します。

```
$ sudo dnf upgrade --security
```

バグ修正プログラムを含めるには、--security オプションを省略します。

Important

- カーネルライブパッチを適用しても、カーネルバージョンは更新されません。バージョンは、インスタンスを再起動した後でのみ新しいバージョンに更新されます。
- AL2023 カーネルは、カーネルライブパッチを 3 か月間入手できます。その後は、そのカーネルバージョンの新しいカーネルライブパッチはリリースされなくなります。
- 3 か月が過ぎた後にカーネルライブパッチを引き続き入手するには、インスタンスを再起動して新しいカーネルバージョンに移行する必要があります。インスタンスは、更新後 3 か月間は引き続きカーネルライブパッチを受け取ります。
- お使いのカーネルバージョンのサポート期間を確認するには、以下のコマンドを実行します。

```
$ 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.

適用されたカーネルライブパッチの表示

適用されたカーネルライブパッチを表示するには

次のコマンドを使用します。

```
$ 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)
```

このコマンドは、ロードおよびインストールされたセキュリティ更新プログラムのカーネルライブパッチのリストを返します。出力例を次に示します。

Note

1 つのカーネルライブパッチには、複数のライブパッチが含まれていてインストールされる場合があります。

カーネルライブパッチの無効化

カーネルライブパッチを使用する必要がなくなった場合は、いつでも無効にできます。

- livepatches の使用の無効化:

1. プラグインの無効化:

```
$ sudo dnf kernel-livepatch manual
```

2. kpatch サービスの無効化:

```
$ sudo systemctl disable --now kpatch.service
```

- livepatch ツールの完全削除:

1. プラグインの削除:

```
$ sudo dnf remove kpatch-dnf
```

2. kpatch-runtime の削除:

```
$ sudo dnf remove kpatch-runtime
```

3. インストールされているすべての livepatches の削除:

```
$ sudo dnf remove kernel-livepatch\*
```

AL2023 での Linux カーネルの更新

トピック

- [AL2023 での Linux カーネルバージョン](#)
- [AL2023 を新しいカーネルバージョンに更新する](#)
- [AL2023 カーネル - よくある質問](#)

AL2023 での Linux カーネルバージョン

AL2023 では、Linux カーネルの長期サポート (LTS) バージョンに基づいて新しいカーネルバージョンが定期的に提供されます。

AL2023 は、もともとカーネル 6.1 で 2023 年 3 月にリリースされました。

2025 年 4 月、AL2023 に Linux カーネル 6.12 のサポートが追加されました。このカーネルでは、EEVDF スケジューリング、FUSE パススルー I/O サポート、新しい Futex API、拡張 eBPF などの新機能が追加されました。また、ユーザースペースのプログラムが、ランタイムにユーザースペースのシャドウスタックとメモリシーリングを用いて自身を保護することも可能になりました。

2026 年 3 月、AL2023 は Linux カーネル 6.18 のサポートを追加しました。更新されたカーネル 6.18 では、プロセッサのサポート、仮想化、セキュリティ、パフォーマンスがさらに向上しました。注目すべき機能には、アーキテクチャ間の IOMMU 機能の改善や、CPU 脆弱性の軽減を管理するための攻撃ベクトル制御などがあります。パフォーマンスの向上は、より高速な FSCRYPT オペレーショ

ンによる暗号化の最適化、メモリ管理の改善、新しいオプトイン、CPU ごとの配列ベースのキャッシュレイヤーとしての Sheaves の導入によって実現されます。

AL2023 を新しいカーネルバージョンに更新する

2026 年 6 月以降、AL2023 はデフォルトのカーネルを毎年更新します。AMIs の [al2023-ami-kernel-default](#) セットは最新の LTS カーネルに更新されるため、新しく起動されたインスタンスは自動的に新しいカーネルバージョンになります。これは、最新のセキュリティ修正とパフォーマンスの向上を最新の状態に保つ最も簡単な方法です。

特定のカーネルバージョンを選択する場合は、目的のカーネルがプリインストールされた AMI を選択するか、既存の AL2023 EC2 インスタンスをアップグレードすることで、カーネル 6.12 または 6.18 で AL2023 を実行できます。

特定のカーネルバージョンで AL2023 AMI を実行する

AWS コンソールを介して、または特定のパラメータについて SSM にクエリを実行して、特定のカーネルがプリインストールされた AL2023 AMI を実行するように選択できます。クエリを実行する SSM キーは `/aws/service/ami-amazon-linux-latest/` で始まり、その後に次のいずれかが続きます。

カーネル 6.12 の場合

- arm64 アーキテクチャの `al2023-ami-kernel-6.12-arm64`
- arm64 アーキテクチャ用 (最小 AMI) の `al2023-ami-minimal-kernel-6.12-arm64`
- x86_64 アーキテクチャの `al2023-ami-kernel-6.12-x86_64`
- x86_64 アーキテクチャ (最小 AMI) の `al2023-ami-minimal-kernel-6.12-x86_64`

カーネル 6.18 の場合

- arm64 アーキテクチャの `al2023-ami-kernel-6.18-arm64`
- arm64 アーキテクチャ用 (最小 AMI) の `al2023-ami-minimal-kernel-6.18-arm64`
- x86_64 アーキテクチャの `al2023-ami-kernel-6.18-x86_64`
- x86_64 アーキテクチャ (最小 AMI) の `al2023-ami-minimal-kernel-6.18-x86_64`

AL2023 AMI の選択の詳細については、「[SSM パラメータとを使用して AL2023 を起動する AWS CLI](#)」を参照してください。

AL2023 インスタンスを新しいカーネルに更新する

以下の手順に従って、実行中の AL2023 インスタンスをカーネル 6.12 または 6.18 にインプレースアップグレードできます。

1. 現在のカーネルを検出し、ターゲットバージョンを設定します。

```
# 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. ターゲットカーネルパッケージをインストールします。

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

3. ターゲットカーネルパッケージの最新バージョンを取得します。

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |
sort -V | tail -1)
```

4. 新しいカーネルをデフォルトのカーネルにします。

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

5. システムを再起動します。

```
$ sudo reboot
```

6. 前のカーネルをアンインストールします。

```
# 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. 追加のカーネルパッケージをターゲットカーネルに相当するものに置き換えます。

```
# Set your target version (change this to your desired kernel: 6.12 or 6.18)
$ TARGET_VERSION="6.12"
```

```
$ for pkg in bpftool kernel-debuginfo kernel-debuginfo-common kernel-headers \
    kernel-modules-extra-common perf python3-perf; do
    case "$pkg" in
        kernel-*)
            src="kernel${SOURCE_VERSION}${pkg#kernel}"
            tgt="kernel${TARGET_VERSION}${pkg#kernel}" ;;
        *)
            src="${pkg}${SOURCE_VERSION}"
            tgt="${pkg}${TARGET_VERSION}" ;;
    esac
    rpm -q "$src" && sudo dnf -y swap "$src" "$tgt"
done

# kernel-tools and kernel-tools-devel must be removed and reinstalled
# together due to version dependency requirements
$ if rpm -q "kernel${SOURCE_VERSION}-tools"; then
    sudo dnf -y remove kernel${SOURCE_VERSION}-tools kernel${SOURCE_VERSION}-tools-
devel
    sudo dnf -y install kernel${TARGET_VERSION}-tools kernel${TARGET_VERSION}-tools-
devel
fi
```

8. (オプション) 以前のカーネルバージョンの kernel-devel をアンインストールします。

```
$ rpm -q kernel${SOURCE_VERSION}-devel && sudo dnf remove -y kernel
${SOURCE_VERSION}-devel
```

以前のカーネルバージョンへのダウングレード

任意の時点で以前のカーネルバージョンにダウングレードする必要がある場合は、次の手順を実行します。

1. 現在のカーネルを検出し、ターゲットバージョンを設定します。

```
# 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. 追加のカーネルパッケージをターゲットカーネルに相当するものに置き換えます。

```
$ for pkg in bpftool kernel-debuginfo kernel-debuginfo-common kernel-headers \
    kernel-modules-extra-common python3-perf; do
    case "$pkg" in
        kernel-*)
            src="kernel${SOURCE_VERSION}${pkg#kernel}"
            tgt="kernel${TARGET_VERSION}${pkg#kernel}" ;;
        *)

```

```

    src="${pkg}${SOURCE_VERSION}"
    tgt="${pkg}${TARGET_VERSION}" ;;
esac
rpm -q "$src" && sudo dnf -y swap "$src" "$tgt"
done

# perf requires --allowerasing due to libtraceevent conflicts
$ rpm -q "perf${SOURCE_VERSION}" && sudo dnf -y swap "perf${SOURCE_VERSION}" "perf
${TARGET_VERSION}" --allowerasing

# kernel-tools and kernel-tools-devel must be removed and reinstalled
# together due to version dependency requirements
$ if rpm -q "kernel${SOURCE_VERSION}-tools"; then
    sudo dnf -y remove kernel${SOURCE_VERSION}-tools kernel${SOURCE_VERSION}-tools-
devel
    sudo dnf -y install kernel${TARGET_VERSION}-tools kernel${TARGET_VERSION}-tools-
devel
fi

```

- ターゲットカーネルパッケージをインストールします。

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

- ターゲットカーネルパッケージの最新バージョンを取得します。

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |
sort -V | tail -1)
```

- ターゲットカーネルをデフォルトのカーネルにします。

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

- システムを再起動します。

```
$ sudo reboot
```

- ソースカーネルをアンインストールします。

```

# 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 カーネル - よくある質問

1. カーネルの更新後には再起動が必要ですか？

実行中のカーネルに変更を加えた場合は必ず再起動が必要です。

2. 複数のインスタンスでカーネルを最新の状態に保つにはどうすればよいですか？

Amazon Linux には、インスタンスのフリートを管理する機能はありません。[AWS Systems Manager](#)などのツールを使用して、大規模なフリートにパッチを適用することをお勧めします。

3. 現在実行しているカーネルのバージョンを確認するにはどうすればよいですか？

AL2023 インスタンスで次のコマンドを実行します。

```
$ uname -r
```

4. AL2023 ではどのカーネルを使用することをお勧めしますか？

最新の AL2023 カーネル 6.18 にアップグレードすることをお勧めしますが、他のすべての AL2023 カーネルは引き続きサポートされています。アップグレードする前にワークロードをテストすることをお勧めします。

5. 既存のアプリケーションは AL2023 カーネルで動作しますか？

AL2023 は、カーネル 6.1 と同じ新しいカーネル (6.12 または 6.18) をサポートしています。アプリケーションが機能し、内部で改善が行われています。いずれの場合も、新しいカーネルに切り替える前に、特定のワークロードをテストする必要があります。

6. カーネル 6.12 または 6.18 のカーネルヘッダー、開発パッケージ、および追加モジュールをインストールするにはどうすればよいですか？

以下を実行してください。

```
$ 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. カーネル 6.12 と 6.18 はどのくらいの期間サポートされますか？

カーネル 6.12 および 6.18 は、2029-06-30 の Amazon Linux 2023 の予定終了までサポートされます。

AL2023 でのランタイムのプログラミングの開始方法

AL2023 では、一部の言語ランタイムについて異なるバージョンが提供されており、AWS は複数のバージョンを同時にサポートする上流プロジェクトと連携しています。名前付きバージョンパッケージのインストール方法や管理方法に関する情報は、これらのパッケージを検索してインストールする `dnf` コマンドを使って検索します。

以下のトピックでは、AL2023 における各言語エコシステムの概要を示しています。

トピック

- [AL2023 での C、C++、および Fortran](#)
- [AL2023 での .NET](#)
- [AL2023 での Go](#)
- [AL2023 での Java](#)
- [AL2023 での Node.js](#)
- [AL2023 での Perl](#)
- [AL2023 での PHP](#)
- [AL2023 での Python](#)
- [AL2023 での Ruby](#)
- [AL2023 での Rust](#)
- [AL2023 での Swift](#)
- [AL2023 での TypeScript](#)

AL2023 での C、C++、および Fortran

AL2023 には GNU コンパイラコレクション (GCC) と LLVM (低レベル仮想化マシン) の Clang フロントエンドの両方が含まれています。

GCC のメジャーバージョンは AL2023 の存続期間を通じて変わりません。マイナーリリースはバグ修正をもたらし、AL2023 リリースに含まれる可能性があります。その他のバグ、パフォーマンス、セキュリティの修正は、AL2023 GCC に含まれているメジャーバージョンにバックポートされる可能性があります。

AL2023 には、C (`gcc`)、C++ (`g++`)、および Fortran (`gfortran`) フロントエンドのデフォルトコンパイラとして GCC のバージョン 11 が含まれています。さらに、AL2023 は、デフォルトバージョ

ンと一緒にインストールできるオプションの代替コンパイラとして GCC バージョン 14 を提供します。

AL2023 では、Ada (gnat)、Go (gcc-go)、Objective-C、Objective-C++ のフロントエンドは有効化されません。

AL2023 RPM のデフォルトのコンパイラフラグには、最適化フラグと強化フラグが含まれます。GCC で独自のコードを構築するには、最適化フラグと強化フラグを含めることをお勧めします。

Note

`gcc --version` が呼び出されると、`gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4)` のようなバージョン文字列が表示されます。Red Hat は Amazon Linux GCC パッケージのベースとなる [GCC ベンダーブランチ](#) を指します。`gcc --help` に示されているバグレポート URL に従い、すべてのバグレポートとサポートリクエストは Amazon Linux に送信する必要があります。

`__GNUC_RH_RELEASE__` マクロなど、このベンダーブランチの長期的な変更の詳細については、「[Fedora パッケージソース](#)」を参照してください。

コアツールチェーンの詳細については、「[コアツールチェーンパッケージglibc、gcc、binutils](#)」を参照してください。

AL2023 および他の Linux ディストリビューションとの関係の詳細については、「[Fedora との関係](#)」を参照してください。

AL2 と比較した AL2023 のコンパイラトリプレットの変更の詳細については、「[コンパイラトリプレット](#)」を参照してください。

トピック

- [GCC 14](#)
- [言語標準バージョンの比較](#)

GCC 14

AL2023 は、デフォルトの GCC 11 と一緒にインストールできるオプションのコンパイラとして GCC 14 を提供しています。GCC 14 には最新の言語機能と最適化が含まれており、新しい C、C++、または Fortran 標準のサポートを必要とするプロジェクトに適しています。

GCC 14 をインストールするには、次のコマンドを使用します。

```
sudo dnf install gcc14 gcc14-c++ gcc14-gfortran
```

GCC 14 コンパイラは、デフォルトの GCC 11 との競合を避けるため、バージョン固有のコマンド名でインストールされます。

- gcc14-gcc - C コンパイラ
- gcc14-g++ - C++ コンパイラ
- gcc14-gfortran - Fortran コンパイラ

使用例:

```
gcc14-gcc -o myprogram myprogram.c
gcc14-g++ -o mycppprogram mycppprogram.cpp
gcc14-gfortran -o myfortranprogram myfortranprogram.f90
```

次のコマンドを実行すると、インストールされているバージョンを確認できます。

```
gcc14-gcc --version
```

これにより、次のようなバージョン情報が表示されます: gcc14-gcc (GCC) 14.2.1 20250110 (Red Hat 14.2.1-7)

Note

GCC 11 と GCC 14 の両方を同じシステムに同時にインストールできます。デフォルトの gcc、g++、gfortran コマンドは引き続き GCC 11 を使用し、バージョン固有のコマンドには GCC 14 を介してアクセスします。

言語標準バージョンの比較

次の表では、さまざまな Amazon Linux バージョンと GCC コンパイラバージョンのデフォルトの言語標準バージョンを比較しています。

Amazon Linux バージョン	C 標準 (デフォルト)	C++ 標準 (デフォルト)	Fortran 標準
AL2 と GCC 7 (デフォルト)	C11 (201112L)	C++14 (201402L)	Fortran 2008
AL2 と GCC 10 (オプション)	C17/C18 (201710L)	C++14 (201402L)	Fortran 2008
AL2023 と GCC 11 (デフォルト)	C17/C18 (201710L)	C++17 (201703L)	Fortran 2008
AL2023 と GCC 14 (オプション)	C17/C18 (201710L)	C++17 (201703L)	Fortran 2008

GCC バージョン別の主な改善点:

- GCC 10 対 GCC 7: デフォルトの C 標準を C11 から C17/C18 にアップグレード、C++20 機能のサポートを追加、最適化機能を改善。
- GCC 11 対 GCC 10: デフォルトの C++ 標準を C++14 から C++17 にアップグレード、C++20 サポートを強化、実験的な C++23 機能を追加。
- GCC 14 対 GCC 11: 完全な C23 標準サポート、C++23 機能の強化、最適化の向上、標準コンプライアンスの向上を追加。

サポートされている言語標準:

- C 標準: すべてのバージョンで C90、C99、C11、C17/C18 をサポート、GCC 10+ で C2x (ドラフト C23) をサポート、GCC 14 で完全な C23 サポートを提供。
- C++ 標準: すべてのバージョンで C++98、C++03、C++11、C++14、C++17、C++20 をサポート、GCC 11+ で実験的な C++23 サポートを提供、GCC 14 で C++23 の機能強化を提供。
- Fortran 標準: すべてのバージョンで主に Fortran 2008 をサポート、GCC バージョンに応じて異なるレベルの Fortran 2018 の機能。

 Note

デフォルトの標準は GCC 11 ~ 14 の間で一貫していますが、GCC 14 では、言語機能のサポートが大幅に改善され、最適化が向上し、診断が強化され、`-std=` フラグを使用して明示的に要求された場合に新しい標準のより完全な実装が提供されます。

AL2023 での .NET

AL2023 は現在、[.NET](#)ランタイムと SDK、バージョン 8.0、9.0、および 10.0 を提供しています。[.NET 6.0](#) は 2024 年 11 月に end-of-life となり、サポートされなくなりました。各バージョンは、アップストリームと同じ期間サポートされます。[.NET](#)。詳細については、「[パッケージのサポートステートメント](#)」を参照してください。

新しい .NET バージョンへの移行

アップストリーム .NET コミュニティは、移行に関する移行ドキュメントを提供します。

- [から .NET 10](#)
- [から .NET 9](#)
- [から .NET 8](#)

AL2023 への .NET のインストール

AL2023 には、複数のバージョンの .NET ランタイムを同時にインストールできます。ただし、パッケージ `dotnet` と `dotnet-host` パッケージは SDK バージョン間で共有されるため、一度に CLI エントリポイントを所有できる SDK `/usr/bin/dotnet` バージョンは 1 つだけです。別の SDK バージョンをインストールすると、共有 `dotnet` コマンドがそのバージョンに更新されます。

特定のバージョンの .NET SDK をインストールするには、`dnf` コマンドを使用します。

```
sudo dnf install dotnet-sdk-10.0
```

を目的のバージョン (8.0、9.0、または 10.0) 10.0 に置き換えます。

SDK なしでランタイムのみをインストールするには:

```
sudo dnf install dotnet-runtime-9.0
```

AL2023 での Go

[Go](#) で記述された独自のコードを Amazon Linux でビルドしたり、AL2023 に含まれるツールチェーンを使用したりできます。AL2 と同様、AL2023 ではオペレーティングシステムの存続期間全体を通じて Go ツールチェーンを更新します。これは、私たちが出荷するツールチェーン内の CVE への対応である場合もあれば、四半期ごとのリリースの一部である場合もあります。

Go は比較的動きの早い言語です。Go で記述された既存のアプリケーションを Go ツールチェーンの新しいバージョンに適応させる必要が生じる場合があります。Go の詳細については、「[Go 1 and the Future of Go Programs](#)」を参照してください。

AL2023 はその存続期間中に新しいバージョンの Go ツールチェーンを組み込む予定ですが、これは上流の Go リリースと歩調を合わせるものではありません。そのため、Go 言語の最先端の機能や標準ライブラリを使用して Go コードをビルドする場合、AL2023 で提供されている Go ツールチェーンの使用は適していない可能性があります。

AL2023 の存続期間中、以前のパッケージバージョンはリポジトリから削除されません。以前の Go ツールチェーンが必要な場合は、新しい Go ツールチェーンのバグ修正やセキュリティ修正を断念し、どの RPM でも利用できるのと同じメカニズムを使用してリポジトリから以前のバージョンをインストールできます。

AL2023 で独自の Go コードを作成したい場合は、AL2023 に含まれている Go ツールチェーンを使用できます。ただし、このツールチェーンは AL2023 の存続期間中に更新される可能性があります。

Go で記述された AL2023 Lambda 関数

Go はネイティブコードにコンパイルされるため、Lambda は Go をカスタムランタイムとして扱います。provided.al2023 ランタイムを使用して、AL2023 で Go 関数を Lambda にデプロイできます。

詳細については、「AWS Lambda デベロッパーガイド」の「[Go による Lambda 関数の構築](#)」を参照してください。

AL2023 での Java

AL2023 には、Java ベースのワークロードをサポートするために複数のバージョンの [Amazon Corretto](#) が用意されています。AL2023 に含まれる Java ベースのパッケージは、すべて Amazon Corretto 17 で構築されています。

Corretto は、Open Java Development Kit (OpenJDK) のビルドで、Amazon による長期サポートが付属しています。Corretto は、Java Technical Compatibility Kit (TCK) を使用した認定によって Java SE 標準に準拠していることが保証されており、Linux、Windows、および macOS で使用できます。

Corretto 1.8.0、Corretto 11、Corretto 17 のそれぞれに [Amazon Corretto](#) パッケージが用意されています。

AL2023 の各 Corretto バージョンは、Corretto バージョンと同じ期間、または AL2023 のサポート終了日のいずれか早い方までサポートされます。詳細については、「[Amazon Linux パッケージのサポートステートメント](#)」および「[Amazon Corretto のよくある質問](#)」を参照してください。

AL2023 での Node.js

[Node.js](#) AL2023 のは、バージョン 20、22、24 で表されます。Amazon Linux はアップストリームの[サポートスケジュール](#)に従い、任意の Node.js バージョンのサポートステータスは、[パッケージのサポートステータスページ](#)で常に確認できます。サポートされているすべての Node.js バージョンには名前空間があり、同じシステムに同時にインストールできます。名前空間を使用すると、Node.js の各インストールがファイルシステム内で一意になります。これは、ランタイムバージョンに基づいてキーディレクトリおよびファイルの名前を変更することで可能になります。実際の実行可能ファイル名は node-{MAJOR_VERSION} または npm-{MAJOR_VERSION} のようになります。ただし、アクティブにできる Node.js バージョンは一度に 1 つだけです。このアクティブなバージョンが、node、npm、/usr/lib/node_modules などのデフォルトのディレクトリとファイル名を提供し、現在アクティブなランタイムを指します。

これを行うには、alternatives ツールの機能を使用します。デフォルトの実行可能ファイル名は仮の名前であり、インストールされている別の Node.js バージョンを指す場合は、いつでも変更される可能性があることに注意してください。この柔軟性により、shebang で node を使用するソフトウェアは、起動時に必要なバージョンを選択できます。ただし、特定のバージョンの Node.js が必要な場合は、指定されたバージョンのランタイムを常に使用する名前空間実行ファイル (node-20 や node-22 など) を呼び出すことで、バージョンの永続化を実現できます。さらに、npm ツールの名前空間付き実行可能ファイル (npm-20 や npm-22 など) は、現在アクティブなランタイムに関係なく、常に対応する Node.js バージョンに関連付けられます。

Node.js は、「」で始まる複数の名前空間パッケージとして配布されます nodejs{MAJOR_VERSION}。これらのパッケージは、ノード、互換性のあるバージョンの npm ツール、ドキュメント、ライブラリなどを提供します。たとえば、22 のノードと npm Node.js は、それぞれ nodejs22 パッケージと nodejs22-npm パッケージによって提供されます。

alternatives ツールには、Node.js バージョンを切り替えるための単一のコマンドが用意されています。デフォルトでは、alternatives は自動モードに設定され、優先度を使用して現在アクティブな Node.js バージョンを決定します。ただし、インストールされているバージョンはどれも随時アクティブ化できます。現在、サポートされている Node.js バージョンの優先度はすべて同じです。つまり、最初にインストールされたバージョンが自動的にアクティブ化されます。

alternatives の便利な使用例

1. どのような alternatives が設定されているかを確認します。

```
alternatives --list
```

2. node の現在の設定を確認する

```
alternatives --display node
```

3. Node.js バージョンをインタラクティブに変更する

```
alternatives --config node
```

4. 手動モードに切り替えて特定のバージョンを選択する

```
alternatives --set node /usr/bin/node-{MAJOR_VERSION}
```

5. バージョンの選択を自動モードに戻す

```
alternatives --auto node
```

セキュリティのベストプラクティス

AL2023 で Node.js と npm を使用する場合は、アップストリームの [Node.js セキュリティのベストプラクティス](#) ガイドに従ってください。サプライチェーン攻撃を含むアプリケーションレベルの脅威や、ロックファイルの使用やライフサイクルスクリプトの無効化などの緩和策について説明します。

AL2023 での Perl

AL2023 は、[Perl](#) プログラミング言語のバージョン 5.32 を提供します。

Perl は過去数十年にわたって Perl 5 リリースの一環として高度な言語互換性を提供してきましたが、AL2023 リリース中に Amazon Linux が Perl 5.32 から移行する予定はありません。Amazon Linux は、[パッケージのサポートステートメント](#)に従い、AL2023 の存続期間中は Perl にセキュリティパッチを適用し続けます。

AL2023 に含まれる Perl モジュール

AL2023 では、さまざまな Perl モジュールが RPM としてパッケージ化されています。RPM として利用できる Perl モジュールは多数ありますが、Amazon Linux では考えられるすべての Perl モジュールをパッケージ化することを目指しているわけではありません。RPM としてパッケージ化されたモジュールは、他のオペレーティングシステム RPM パッケージに依存している可能性があるため、Amazon Linux は単なる機能更新よりもこうしたセキュリティパッチの適用を優先します。

AL2023 には、Perl デベロッパーが Perl モジュール用の慣用的なパッケージマネージャーを使用できるようにするために、CPAN も含まれています。

AL2023 での PHP

AL2023 は現在、[PHP](#) プログラミング言語のバージョン 8.1、8.2、8.3、8.4、8.5 を提供しています。各バージョンは、アップストリームの PHP と同じ期間サポートされます。詳細については、「[パッケージのサポートステートメント](#)」を参照してください。

古い PHP バージョンからの移行

アップストリームの PHP コミュニティでは、以下の移行に関する包括的な移行ドキュメントをまとめています。

- [8PHP.4.x から PHP 8.5.x へ](#)
- [PHP 8.3.x から PHP 8.4.x](#)
- [PHP 8.2.x から PHP 8.3.x](#)
- [PHP 8.1.x から PHP 8.2.x](#)
- [PHP 8.0.x から PHP 8.1.x](#)

AL2 には amazon-linux-extras に PHP 8.0、8.1、および 8.2 が組み込まれており、AL2023 に簡単にアップグレードできます。

PHP 7.x バージョンからの移行

Note

PHP プロジェクトでは、[サポートされているバージョン](#)のリストとスケジュール、および[サポートされていないブランチ](#)のリストを管理しています。

AL2023 がリリースされたとき、[PHP](#) の 7.x バージョンと 5.x バージョンはどれも PHP コミュニティによってサポートされておらず、AL2023 のオプションとしても含まれていませんでした。

アップストリームの PHP コミュニティでは、[PHP 7.4 から PHP 8.0 への移行に関する包括的な移行ドキュメント](#)をまとめています。PHP 8.1 と PHP 8.2 への移行に関する前のセクションで参照したドキュメントと組み合わせることで、PHP ベースのアプリケーションを最新の PHP に移行することができます。

Note

AL2 には、amazon-linux-extras に PHP 7.1、7.2、7.3、および 7.4 が含まれています。これらの追加はすべてサポート終了であり、今後のセキュリティ更新は保証されていないことに注意してください。

AL2023 の PHP モジュール

AL2023 には、PHP コアに組み込まれた多くの PHP モジュールが含まれています。AL2023 は、すべてのパッケージを [PHP 拡張コミュニティライブラリ \(PECL\)](#) に含めることを目指しているわけではありません。

AL2023 での Python

AL2023 では Python 2.7 が削除され、Python を必要とするコンポーネントはすべて Python 3 で動作するように記述されています。

AL2023 では、AL2023 に含まれている Python コードと同様に、カスタマーコードとの互換性を保持するために Python 3 を `/usr/bin/python3` として使用可能な状態にしています。AL2023 の存続期間中、これは Python 3.9 のままになります。

`/usr/bin/python3` が指している Python のバージョンはシステム Python と見なされ、AL2023 の場合は Python 3.9 です。

AL2023 は現在、バージョン 3.11Python、3.12、3.13、3.14 など、の新しいバージョンを提供しています。各バージョンは、アップストリームと同じ期間サポートされますPython。詳細については、「[パッケージのサポートステートメント](#)」を参照してください。

AL2023 Python では複数のバージョンを同時にインストールできます。`/usr/bin/python3` は常に Python 3.9 ですが、Python の各バージョンには名前空間があり、バージョン番号で確認できます。例えば、python3.11 がインストールされている場合、`/usr/bin/python3.11` は `/usr/bin/python3.9` および `/usr/bin/python3.9` への `/usr/bin/python3` シンボリックリンクとともに存在します。

Note

`/usr/bin/python3` シンボリックリンクが指すものは変更しないでください。変更すると AL2023 のコア機能に問題が発生する可能性があります。

新しいPythonバージョンへの移行

アップストリームPythonコミュニティは、移行に関する移行ドキュメントを提供します。

- [から Python 3.14](#)
- [から Python 3.13](#)
- [から Python 3.12](#)
- [から Python 3.11 へ](#)

AL2023 の Python モジュール

AL2023 では、さまざまな Python モジュールが RPM としてパッケージ化されています。通常、Python モジュールの RPM は Python のシステムバージョンのみをターゲットに構築されます。

AL2023 での Ruby

[Ruby](#) AL2023 の は、バージョン 3.2 および 3.4 で表されます。Amazon Linux はアップストリームのサポートスケジュールに従い、どのRubyバージョンのサポートステータスも [Ruby ウェブサ](#)

[イト](#)でいつでも確認できます。サポートされているすべての Ruby バージョンには名前空間があり、同じシステムに同時にインストールできます。名前空間を使用すると、Ruby の各インストールがファイルシステム内で一意になります。これは、ランタイムバージョンに基づいてキーディレクトリおよびファイルの名前を変更することで可能になります。実際の実行可能ファイル名は `ruby{MAJOR.MINOR}` (例: `ruby3.2` または `ruby3.4`) のようになります。Ruby 3.4 は、MRI (Matz の Interpreter) Ruby 名前空間バイナリ も提供します。これは `ruby3.4-mri`、の標準の C ベースのリファレンス実装を参照します Ruby。ただし、アクティブにできる Ruby バージョンは一度に 1 つだけです。このアクティブなバージョンは、`ruby`、`gem`、バンドルなどのデフォルトのディレクトリとファイル名を提供し、現在アクティブなランタイムを指します。

これを行うには、`alternatives` ツールの機能を使用します。デフォルトの実行可能ファイル名は仮の名前であり、インストールされている別の Ruby バージョンを指す場合は、いつでも変更される可能性があることに注意してください。この柔軟性により、シェバンで `ruby` を使用して呼び出されたときに目的のバージョンを選択するソフトウェアが可能になります。ただし、特定のバージョンの Ruby が必要な場合、指定されたバージョンのランタイムを常に使用する名前空間付き実行可能ファイル (`ruby3.2` や など `ruby3.4`) を呼び出すことで、バージョンの永続性を実現できます。さらに、`ruby3.2-gem`、`ruby3.4-gem` `ruby3.2-bundler` などの `gem` およびバンドルツールの名前空間付き実行可能ファイルは `ruby3.4-bundler`、現在アクティブなランタイムに関係なく、常に対応する Ruby バージョンに関連付けられます。

Ruby は、「`r`」で始まる複数の名前空間パッケージとして配布されます `ruby{MAJOR.MINOR}`。これらのパッケージは、Gem およびバンドルツール、ドキュメント、ライブラリなどの `ruby` 互換バージョンを提供します。たとえば、コア Ruby 3.2 ランタイムは `ruby3.2` パッケージによって提供され、依存関係として (`gem ruby3.2-rubygems` の提供) と `ruby3.2-rubygem-bundler` (バンドルとバンドルの提供) を取り込みます。

Ruby バージョンをインストールした後、代替設定でコンパニオンツールのエントリが `null` と表示される場合があります。これは、`alternatives --display ruby` を実行して検証できます。エントリが `null` と表示される場合は、`alternatives --install` を使用して手動で登録する必要があります。たとえば、3.4 Ruby のすべてのコンパニオンツールを登録するには:

```
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
```

優先度値 (例: 3.4 の場合は Ruby 34、3.2 の場合は Ruby 32) は、メインの ruby 代替エントリで使われる優先度と一致する必要があります。登録すると、コンパニオンツールは ruby の代替手段とともに自動的に管理されます。

alternatives ツールには、Ruby バージョンを切り替えるための単一のコマンドが用意されています。デフォルトでは、alternatives は自動モードに設定され、優先度を使用して現在アクティブな Ruby バージョンを決定します。ただし、インストールされているバージョンはどれも随時アクティブ化できます。現在、サポートされている Ruby バージョンの優先度はすべて同じです。つまり、最初にインストールされたバージョンが自動的にアクティブ化されます。

alternatives の便利な使用例

1. どのような alternatives が設定されているかを確認します。

```
alternatives --list
```

2. Ruby の現在の設定を確認する

```
alternatives --display ruby
```

3. Ruby バージョンをインタラクティブに変更する

```
alternatives --config ruby
```

4. 手動モードに切り替えて特定のバージョンを選択する

```
alternatives --set ruby /usr/bin/ruby{MAJOR.MINOR}
```

5. バージョンの選択を自動モードに戻す

```
alternatives --auto ruby
```

AL2023 での Rust

[Rust](#) で記述されたコードを Amazon Linux でビルドしたり、AL2023 に含まれるツールチェーンを使用したりできます。

AL2 と同様、AL2023 ではオペレーティングシステムの存続期間全体を通じて Rust ツールチェーンを更新します。これは、私たちが出荷するツールチェーン内の CVE への対応である場合もあれば、四半期ごとのリリースの一部である場合もあります。

[Rust](#) は比較的動きの速い言語であり、約 6 週間の間隔で新しいリリースが行われます。これらのリリースで、新しい言語や標準ライブラリ機能が追加される可能性があります。AL2023 はその存続期間中に新しいバージョンの Rust ツールチェーンを組み込む予定ですが、これは上流の Rust リリースと歩調を合わせるものではありません。そのため、Rust 言語の最先端の機能を使用して Rust コードをビルドする場合、AL2023 で提供されている Rust ツールチェーンの使用は適していない可能性があります。

AL2023 の存続期間中、古いパッケージバージョンはリポジトリから削除されません。古い Rust ツールチェーンが必要な場合は、新しい Rust ツールチェーンのバグ修正やセキュリティ修正を断念し、どの RPM でも利用できるのと同じメカニズムを使用してリポジトリから古いバージョンをインストールできます。

AL2023 で独自の Rust コードを作成したい場合は、AL2023 に含まれている Rust ツールチェーンを使用できます。ただし、このツールチェーンは AL2023 の存続期間中に更新される可能性があります。

Rust で記述された AL2023 Lambda 関数

Rust はネイティブコードにコンパイルされるため、Lambda は Rust をカスタムランタイムとして扱います。provided.al2023 ランタイムを使用して、AL2023 で Rust 関数を Lambda にデプロイできます。

詳細については、「AWS Lambda デベロッパーガイド」の「[Rust による Lambda 関数の構築](#)」を参照してください。

AL2023 での Swift

AL2023 は現在、[Swift](#) ランタイムと SDK を提供しています。

AL2023 では、SwiftSDK とランタイムは別々のパッケージに分割されます。swiftlang-lib パッケージにはランタイムライブラリが含まれ、swiftlang パッケージには完全な SDK が含まれています。これにより、ユーザーは完全な Swift SDK をインストールしなくても、本番ホストで SDK でコンパイルされたバイナリを実行できます。現時点では、異なるバージョンの Swift には ABI との互換性がないため、更新するときは Swift、アプリケーションをデプロイする前に新しいバージョンで再構築する必要があります。

Note

と同様に Rust、Swift バージョンは Swift プロジェクトの最新の安定バージョンに従います。パッケージは名前空間化されず、一度に 1 つのバージョンのみがサポートされます。

AL2023 Swiftでの の使用に関する注意事項

- C++ 相互運用性 – システムのパッケージマネージャーは と連携しない clang 場合があります Swift。これを解決するには、swift-clang を に追加する PATH が、代替方法を使用して clang を に設定します swift-clang。
- Swift コードのデバッグ – システムは Swift ソースをデバッグ LLDB できません。Swift インストールディレクトリを に追加する PATH が、代替手段を使用して lldb を に設定することで、Swift パッケージ LLDB に含まれている を使用します swift-lldb。

AL2023 での TypeScript

Note

このドキュメントでは、TypeScript とその Node.js ベースの実行環境に関する重要な情報を提供します。また、一般的な開発ワークフローを紹介し、一貫性と再現性のある開発環境を提供するために TypeScript が AL2023 にどのようにパッケージ化されているかを説明します。

[TypeScript](#) (TS) は、JavaScript (JS) に基づくプログラミング言語であり、JS のすべての機能を提供するとともに、[タイプシステムで拡張](#)されています。一般的なシナリオでは、TS で記述されたプログラムは最初に JS コードに変換され、次に Node.js によって他の通常の JS プログラムと同様に実行されます。TS の専門用語では、この変換プロセスを「[コンパイル](#)」、これを実行するのが「[コンパイラ](#)」(tsc)と呼ばれています。tsc コンパイラ自体は JS で記述されているため、実行するには Node.js などの JS ランタイム環境も必要です。他の JS ランタイム環境とは異なり、Node.js は現在、実験的で軽量の TS のみをサポートしています。タイプチェックを含む完全な TS サポートでは、[typescript](#) などのサードパーティ製パッケージを使用する必要があります。Node.js ランタイム環境の tsc (TS コンパイラ) を取得するには、typescript ノードモジュールをインストールするのが一般的です。これを行うには、パッケージマネージャーの 1 つ (通常は npm) を使用できます。npm を使用して TS コンパイラをインストールするには、グローバルとプロジェクト内の 2 つの方法が

あります。[正式に推奨されているインストール方法](#)は、プロジェクトごとに TS コンパイラをインストールすることであり、プロジェクトの長期的な一貫性と再現性が確保されます。ただし、TS コンパイラをグローバルにインストールすることは、ホスト全体とその JS ランタイムに同じバージョンを提供するため、TS コンパイラがローカルにインストールされていないプロジェクトにも役立つ場合があります。これは、Amazon Linux で利用可能な RPM パッケージ (nodejs20-typescript や nodejs22-typescript など) では、まさにこの方法により、TS コンパイラをシステムレベルでグローバルに、かつサポートされている Node.js バージョンごとに個別にインストールします。

tsc はどの Node.js バージョンにも直接依存しません。コンパイラは、[target](#) や [lib](#) などのオプションを介して、特別なファイル (tsconfig.json) に定義されている特定レベルのランタイム機能を必要とします。これらのオプションの値は、[ECMAScript](#) (ES) 標準のバージョンを表し、JS ランタイム環境でサポートされる場合とそうでない場合があります。Node.js のバージョンごとにサポートされる ES 標準のバージョンは異なります。Node.js のバージョンが新しいほど、サポートされる ES 標準バージョンはより高く、より完全になります。tsconfig.json がプロジェクトのルートディレクトリに存在しない場合は、デフォルトの設定オプションセットが使用されます。Node.js のさまざまなバージョンと、さまざまな ES 標準バージョンでサポートされている機能との互換性の表は、[node.green](#) で参照できます。tsc には 100 を超えるさまざまなオプションがあり、tsconfig.json に定義されている場合があります。設定の連鎖もサポートされており、別のファイルで定義した設定オプションをメインファイルに含めることができます。このアプローチにより、特定バージョンの Node.js と互換性のある [Base TS Config](#) をインストールし、これをプロジェクト固有のオプションで拡張できます。幸い、Node.js の Base TS Configs は、ノードモジュールとして提供されており、npm を使用してプロジェクトフォルダにインストールできます。Node.js バージョン [18](#)、[20](#)、[22](#) で config のソースコードを参照してください。

Node.js ベースのランタイム設計には特定の弱点があります。ホストでサポートされるランタイムのバージョンは 1 つだけで、プロジェクトレベルですべての依存関係の再現性と一貫性が求められることです。これにより、TypeScript を使用する一般的なアプローチとして、TS コンパイラ、Node.js 現行バージョンの TS 基本設定、およびすべてのソフトウェアの依存関係をプロジェクト内にローカルにインストールしています。グローバルにインストールされるノードモジュールは npm などの CLI ツールのみであることが予想されますが、同じく CLI ツールである tsc がグローバルにインストールされることはほとんどありません。幸い、tsc のグローバル (システム全体) インストールとローカル (プロジェクト内) インストールは問題なく共存でき、異なるバージョンとして個別に使用することもできます。ローカルにインストールした tsc は、npx ツールを使用して実行する必要があることに注意してください。このツールは npm と一緒にインストールされます。したがって、システム TS コンパイラを使用する場合でも、ユーザーはランタイムのコンポーネントのバージョンを選択できます。例えば、Node.js (アクティブバージョンを代替手段で切り替える)、TS コンパイラ

(ローカルまたはグローバルにインストールし、アクティブバージョンを代替手段で切り替える) を選択し、特定のニーズに合わせて設定できます。

Amazon Linux は、npm などのグローバルにインストールされた他のノードモジュールと同じ方法で、TS コンパイラを Node.js バージョンごとにパッケージ化します。パッケージとバイナリは名前空間化され、名前の一部として Node.js のメジャーバージョンが含まれます。コンパイラのデフォルトの実行可能ファイル名 `tsc` は、`alternatives` ツールによってランタイムに管理され、ファイルがインストールされて実行される Node.js の現在アクティブなバージョンを指します。この選択は、現在の Node.js ランタイムバージョンには依存しません。ノードの実行可能ファイルで Node.js 20 を指し、`tsc` を Node.js 22 で解釈するように設定することもできます。TS コンパイラの名前空間名を使用することもできます。例えば、`tsc-{MAJOR_VERSION}` を、デフォルトの `tsc` 名が設定されている名前空間と独立して使用できます。

TS コンパイラのアクティブバージョンを管理するために役立つコマンド

1. どのような `alternatives` が設定されているかを確認します。

```
alternatives --list
```

2. `tsc` の現在の設定を確認します。

```
alternatives --display tsc
```

3. `tsc` バージョンをインタラクティブに変更します。

```
alternatives --config tsc
```

4. 手動モードに切り替えて特定のバージョンを選択します。

```
alternatives --set tsc /usr/bin/tsc-{MAJOR_VERSION}
```

5. 自動バージョンの選択モードに戻ります。

```
alternatives --auto tsc
```

同じシステムに複数のバージョンの Node と TS コンパイラをインストールして使用する例:

```
# 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 予約済みユーザーおよびグループ

AL2023 は、イメージのプロビジョニング中と特定のパッケージのインストール中に、特定のユーザーとグループを事前に割り当てます。競合を防ぐために、ユーザー、グループ、および関連する UID と GID の一覧をここに示します。

トピック

- [AL2023 の予約済みユーザーのリスト](#)
- [AL2023 リザーブドグループのリスト](#)

AL2023 の予約済みユーザーのリスト

ユーザー名	UID
ルート	0
bin	1
daemon	2
adm	3
lp	4
sync	5
shutdown	6
停止	7
メール	8
オペレーター	11
games	12
ftp	14

ユーザー名	UID
squid	23
named	25
postgres	26
MySQL	27
nscd	28
nscd	28
rpcuser	29
rpc	32
mailnull	47
apache	48
smmsp	51
tomcat	53
ldap	55
tss	59
nslcd	65
avahi	70
tcpdump	72
sshd	74
radvd	75
dbus	81

ユーザー名	UID
postfix	89
dovecot	97
stapusr	156
stapsys	157
stapdev	158
avahi-autoipd	170
pulse	171
rtkit	172
sanlock	179
systemd-network	192
systemd-resolve	193
uidd	961
stap-server	962
systemd-journal-remote	963
redis6	970
pesign	971
smtpq	972
smtpd	973
nginx	974
munge	975

ユーザー名	UID
memcached	976
sphinx	977
haproxy	978
flatpak	979
debuginfod	980
dovnull	981
dnsmasq	982
unbound	983
clamscan	984
clamilt	985
clamupdate	986
colord	987
ods	988
aws-kinesis-agent-user	989
saslauth	990
CWAgent	991
polkitd	992
ec2-instance-connect	993
chrony	994
systemd-timesync	995

ユーザー名	UID
systemd-coredump	996
libstoragemgmt	997
systemd-oom	999
ec2-user	1,000
nobody	65534

名前順に一覧表示

ユーザー名	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

ユーザー名	UID
daemon	2
dbus	81
debuginfod	980
dnsmasq	982
dovecot	97
dovnull	981
ec2-instance-connect	993
ec2-user	1,000
flatpak	979
ftp	14
games	12
停止	7
haproxy	978
ldap	55
libstoragemgmt	997
lp	4
メール	8
mailnull	47
memcached	976
munge	975

ユーザー名	UID
MySql	27
named	25
nginx	974
nobody	65534
nscd	28
nscd	28
nsld	65
ods	988
オペレーター	11
pesign	971
polkitd	992
postfix	89
postgres	26
pulse	171
radvd	75
redis6	970
ルート	0
rpc	32
rpcuser	29
rtkit	172

ユーザー名	UID
sanlock	179
saslauth	990
shutdown	6
smmsp	51
smtpd	973
smtpq	972
sphinx	977
squid	23
sshd	74
stap-server	962
stapdev	158
stapsys	157
stapusr	156
sync	5
systemd-coredump	996
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995

ユーザー名	UID
tcpdump	72
tomcat	53
tss	59
unbound	983
uidd	961

AL2023 リザーブドグループのリスト

グループ名	GID
root	0
bin	1
daemon	2
sys	3
adm	4
tty	5
disk	6
disk	6
lp	7
mem	8
kmem	9
wheel	10

グループ名	GID
cdrom	11
mail	12
mail	12
man	15
dialout	18
floppy	19
games	20
slocate	21
utmp	22
squid	23
named	25
postgres	26
MySQL	27
nscd	28
nscd	28
rpcuser	29
rpc	32
tape	33
utempter	35
kvm	36

グループ名	GID
video	39
mailnull	47
apache	48
ftp	50
smmsp	51
tomcat	53
lock	54
ldap	55
tss	59
audio	63
avahi	70
tcpdump	72
sshd	74
radvd	75
saslauth	76
dbus	81
screen	84
wbpriv	88
postfix	89
postdrop	90

グループ名	GID
dovecot	97
users	100
input	104
render	105
sgx	106
mock	135
stapusr	156
stapusr	156
stapsys	157
stapsys	157
stapdev	158
stapdev	158
avahi-autoipd	170
pulse	171
rtkit	172
sanlock	179
systemd-journal	190
systemd-network	192
systemd-resolve	193
usbmon	959

グループ名	GID
wireshark	960
uidd	961
stap-server	962
systemd-journal-remote	963
usershares	964
redis6	965
pesign	966
smtpq	967
smtpd	968
nginx	969
munge	970
memcached	971
sphinx	972
tracing	973
haproxy	974
flatpak	975
debuginfod	976
dovnull	977
dnsmasq	978
unbound	979

グループ名	GID
clamscan	980
clamilt	981
virusgroup	982
virusgroup	982
virusgroup	982
clamupdate	983
printadmin	984
colord	985
ods	986
docker	987
aws-kinesis-agent-user	988
cwagent	989
pulse-rt	990
pulse-access	991
ec2-instance-connect	993
chrony	994
systemd-timesync	995
systemd-coredump	996
libstoragemgmt	997
ssh_keys	998

グループ名	GID
systemd-oom	999
ec2-user	1000
ne	1001
polkitd	9920
nobody	65534

名前順に一覧表示

グループ名	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

グループ名	GID
colord	985
cwagent	989
daemon	2
dbus	81
debuginfod	976
dialout	18
disk	6
disk	6
dnsmasq	978
docker	987
dovecot	97
dovnull	977
ec2-instance-connect	993
ec2-user	1000
flatpak	975
floppy	19
ftp	50
games	20
haproxy	974
input	104

グループ名	GID
kmem	9
kvm	36
ldap	55
libstoragemgmt	997
lock	54
lp	7
mail	12
mail	12
mailnull	47
man	15
mem	8
memcached	971
mock	135
munge	970
MySql	27
named	25
ne	1001
nginx	969
nobody	65534
nscd	28

グループ名	GID
nscd	28
ods	986
pesign	966
polkitd	9920
postdrop	90
postfix	89
postgres	26
printadmin	984
pulse	171
pulse-access	991
pulse-rt	990
radvd	75
redis6	965
render	105
root	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179
saslauth	76

グループ名	GID
screen	84
sgx	106
slocate	21
smmsp	51
smtpd	968
smtpq	967
sphinx	972
squid	23
ssh_keys	998
sshd	74
stap-server	962
stapdev	158
stapdev	158
stapsys	157
stapsys	157
stapusr	156
stapusr	156
sys	3
systemd-coredump	996
systemd-journal	190

グループ名	GID
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
tape	33
tcpdump	72
tomcat	53
tracing	973
tss	59
tty	5
unbound	979
usbmon	959
users	100
usershares	964
utempter	35
utmp	22
uudd	961
video	39
virusgroup	982

グループ名	GID
virusgroup	982
virusgroup	982
wbpriv	88
wheel	10
wireshark	960

AL2023 で利用可能なコーデックのリスト

AL2023 は、標準リポジトリを通じてさまざまなマルチメディアコーデックを提供します。このページでは、コーデックとその一般的なユースケースの概要を説明します。

Important

Amazon Linux に含まれるコーデックの使用と配布には、特定のサードパーティーのオーディオ形式やビデオ形式の所有者またはライセンサーなど、サードパーティーからライセンス権限の取得が必要になる場合があります。これらのライセンスの取得と必要な使用料または料金の支払いについては、お客様が全責任を負うものとします。

コーデック	説明
flac	データや品質を失うことなくオーディオを圧縮する無料のオープンソースのロスレスオーディオコーデック。高品質のオーディオストレージで一般的に使用されます。
fdk-aac-free	AAC (Advanced Audio Codec) 標準のオープンソース実装。ストリーミングやファイルストレージなどの MP3 の代わりとして高品質のオーディオ圧縮を提供します。
webrtc-audio-processing	WebRTC (Web Real-Time Communication) で使用されるオーディオ処理用のライブラリ。ノイズ抑制、エコーキャンセル、ゲインコントロールなどの機能を提供します。
opus	リアルタイムのストリーミング用に設計された、汎用性が高く効率的なオーディオコーデック。低レイテンシーで、VoIP や音楽ストリーミングなどの幅広いオーディオアプリケーションをサポートします。

コーデック	説明
libsndfile	さまざまな形式 (WAV、AIFF、FLAC など) のオーディオファイルを読み書きするためのライブラリ。オーディオの処理および操作ツールで一般的に使用されます。
openh264	H.264 ビデオコーデックのオープンソース実装。ストリーミング、会議、マルチメディアアプリケーションで使用される H.264 ビデオのエンコードとデコードのサポートを提供します。
svt-av1	AV1 ビデオコーデックのオープンソースの高性能実装。AV1 ビデオのスケラブルなエンコーディングとデコードを提供し、最新の CPUs。
dav1d	速度、効率、移植性に焦点を当てた AV1 ビデオデコーダーのオープンソース実装。幅広いプラットフォームとアプリケーションに高性能 AV1 デコードを提供します。
libde265	H.265/HEVC ビデオデコーダーのオープンソース実装。マルチメディアアプリケーションやフレームワークでできるように、HEVC ビデオストリームを効率的かつポータブルにデコードします。
libheif	HEIF および AVIF イメージファイルを読み書きするためのオープンソースライブラリ。HEVC や AV1 などの最新の圧縮形式を使用して、イメージとイメージシーケンスの効率的なエンコード、デコード、変換を提供します。
mpg123	MP3 ファイルを含む MPEG オーディオストリーム用の高性能デコーダー。再生、ストリーミング、オーディオ処理アプリケーションに高速で正確なオーディオデコードを提供します。

コーデック	説明
x265	x265 の主な目的は、利用可能な最高の H.265/HEVC エンコーダーになり、さまざまなハードウェアプラットフォームで最高の圧縮効率と最高のパフォーマンスを提供することです。このパッケージには、コマンドラインエンコーダーが含まれています。

Amazon Linux 2023 でのセキュリティおよびコンプライアンス

Important

脆弱性を報告する場合、または AWS クラウドサービスやオープンソースプロジェクトに関するセキュリティ上の懸念がある場合は、[「脆弱性レポート」ページ](#)を使用して AWS セキュリティにお問い合わせください。

でのクラウドセキュリティが最優先事項 AWS です。お客様は AWS、セキュリティを最も重視する組織の要件を満たすように構築されたデータセンターとネットワークアーキテクチャを活用できます。

セキュリティは、AWS とお客様の間の責任共有です。[責任共有モデル](#)ではこれをクラウドのセキュリティおよびクラウド内のセキュリティと説明しています。

- クラウドのセキュリティ – AWS クラウドで AWS サービスを実行するインフラストラクチャを保護する AWS 責任があります。AWS また、では、安全に使用できるサービスも提供しています。サードパーティーの監査者は、[AWS コンプライアンスプログラム](#)コンプライアンスプログラムの一環として、当社のセキュリティの有効性を定期的にテストおよび検証。AL2023 に適用されるコンプライアンスプログラムの詳細については、「[コンプライアンスプログラムAWS による対象範囲内のサービスコンプライアンスプログラム](#)」を参照してください。
- クラウド内のセキュリティ – お客様の責任は使用する AWS のサービスによって決まります。また、お客様は、お客様のデータの機密性、企業の要件、および適用可能な法律および規制などの他の要因についても責任を担います。

トピック

- [AL2023 に関する Amazon Linux セキュリティアドバイザリ](#)
- [適用可能なアドバイザリの一覧表示](#)
- [セキュリティ更新プログラムのインプレースでの適用](#)
- [AL2023 の SELinux モードの設定](#)
- [AL2023 で FIPS モードを有効にする](#)
- [AL2023 コンテナで FIPS モードを有効にする](#)

- [AL2023 で OpenSSL FIPS プロバイダーを切り替える](#)
- [AL2023 カーネル強化](#)
- [AL2023 でのリポジトリメタデータ署名](#)
- [AL2023 での UEFI セキュアブート](#)

AL2023 に関する Amazon Linux セキュリティアドバイザリ

Amazon Linux の安全性を保つため、懸命に取り組んでいますが、修正が必要なセキュリティ上の問題が発生することがあります。アドバイザリは、修正が入手可能になると公開されます。アドバイザリが公開される主な場所は、Amazon Linux Security Center (ALAS) のウェブサイトです。詳細については「[Amazon Linux Security Center](#)」を参照してください。

Important

脆弱性を報告する場合、または AWS クラウドサービスやオープンソースプロジェクトに関するセキュリティ上の懸念がある場合は、「[脆弱性レポート](#)」ページを使用して AWS セキュリティにお問い合わせください。

AL2023 に影響する問題や関連する更新に関する情報は、Amazon Linux チームが複数の場所で公開します。セキュリティツールでは、これらの主要な情報源から情報を取得し、その結果をユーザーに提示するのが一般的です。このため、Amazon Linux が公開するプライマリソースに直接触れずに、[Amazon Inspector](#) などの希望するツールが提供するインターフェイスで確認することになる可能性があります。

Amazon Linux Security Center の発表

Amazon Linux の発表は、アドバイザリには該当しない事項に関して提供されます。このセクションでは、ALAS 自体に関する発表と、アドバイザリに当てはまらない情報について説明します。詳細については「[Amazon Linux Security Center \(ALAS\) Announcements](#)」を参照してください。

例えば、「[2021-001 - Amazon Linux Hotpatch Announcement for Apache Log4j](#)」は、アドバイザリではなく発表に該当します。ここでは、Amazon Linux に、Amazon Linux に含まれていないソフトウェアのセキュリティ問題の軽減に役立つパッケージが追加されたことを発表しています。

[Amazon Linux Security Center CVE Explorer](#) も、ALAS の発表で公開されました。詳細については、「[New website for CVEs](#)」を参照してください。

Amazon Linux Security Center のよくある質問

ALAS および Amazon Linux による CVE の評価方法に関するよくある質問への回答については、[Amazon Linux Security Center \(ALAS\) Frequently Asked Questions \(FAQs\)](#)」を参照してください。

ALAS アドバイザリ

Amazon Linux Advisory は、Amazon Linux ユーザーに関連する重要な情報、特にセキュリティ更新に関する情報を提供します。[Amazon Linux Security Center](#) は、アドバイザリがウェブ上で公開される場所です。アドバイザリ情報は、RPM パッケージリポジトリのメタデータの一部としても提供されます。

アドバイザリと RPM リポジトリ

Amazon Linux 2023 パッケージリポジトリには、0 件以上の更新を記述するメタデータが含まれている場合があります。dnf updateinfo コマンドの名前は、この情報を含むリポジトリメタデータファイル名 updateinfo.xml にちなんで命名されています。コマンドの名前は updateinfo で、メタデータファイルは update を指しますが、これらはすべてアドバイザリの一部であるパッケージの更新を参照しています。

Amazon Linux Advisories は、dnf パッケージマネージャーが参照する RPM リポジトリのメタデータに存在する情報とともに、[Amazon Linux Security Center](#) のウェブサイトで公開されます。ウェブサイトとリポジトリのメタデータは最終的に一貫性が維持されますが、ウェブサイト上の情報とリポジトリのメタデータが一致しない場合もあります。これは通常、AL2023 の新しいリリースがリリース途上で、最新の AL2023 リリース後にアドバイザリが更新される場合に発生します。

通常、問題に対処するパッケージ更新と同時に新しいアドバイザリが発表されますが、必ずしもそうであるとは限りません。アドバイザリは、リリース済みのパッケージで対処されている新しい問題に関して作成される場合もあります。既存のアドバイザリが、既存の更新で対処された新しい CVE で更新される場合もあります。

Amazon Linux 2023 の「[AL2023 でのバージョン管理されたリポジトリを介した確定的なアップグレード](#)」機能は、特定の AL2023 バージョンの RPM レポジトリに、このバージョンの時点での RPM レポジトリのメタデータスナップショットが含まれていることを意味します。これには、セキュリティ更新を記述するメタデータが含まれます。特定の AL2023 バージョンの RPM リポジトリは、リリース後には更新されません。AL2023 RPM リポジトリの古いバージョンを参照する場合、新規または更新されたセキュリティアドバイザリは表示されません。dnf パッケージマネージャー

を使用してlatestリポジトリバージョンまたは特定の AL2023 リリースを確認する方法については、「[適用可能なアドバイザリの一覧表示](#)」セクションを参照してください。

Advisory ID

各アドバイザリは id で参照されます。現時点での Amazon Linux の傾向として、[Amazon Linux Security Center](#) ウェブサイトでは、アドバイザリを「[ALAS-2024-581](#)」として表示し、dnf パッケージマネージャーは、[同じアドバイザリを「ALAS2023-2024-581」の ID を持つものとして表示する](#)という点があります。[セキュリティ更新プログラムのインプレースでの適用](#)場合、特定のアドバイザリを参照するには、パッケージマネージャー ID を使用する必要があります。

Amazon Linux では、OS のメジャーバージョンごとに独自の Advisory ID の名前空間が存在します。Amazon Linux Advisory ID の形式については特定の想定をすべきではありません。これまで、Amazon Linux Advisory ID は、NAMESPACE-YEAR-NUMBER というパターンに従っていました。NAMESPACE に指定可能な値の全範囲は定義されていませんが、ALAS、ALASCORRETT08、ALAS2023、ALAS2、ALASPYTHON3.8、ALASUNBOUND-1.17 などがあります。YEAR はアドバイザリが作成された年、NUMBER は名前空間内での一意の整数です。

Advisory ID は通常、更新のリリース順での連番ですが、そうならない理由も数多くあり、必ずしも連番になるとは限らないため、連番であると想定すべきではありません。

Advisory ID は、Amazon Linux の各メジャーバージョンに固有の、非透過的な文字列として扱います。

Amazon Linux 2 では、各 Extra は個別の RPM リポジトリに格納され、アドバイザリのメタデータは関連するリポジトリ内にのみ格納されています。あるリポジトリのアドバイザリが、別のリポジトリに適用されることはありません。現時点では、[Amazon Linux Security Center](#) ウェブサイトでは、主要な Amazon Linux バージョンごとにアドバイザリリストが 1 つずつ提供されており、リポジトリごとのリストには分割されていません。

AL2023 は、パッケージの代替バージョンのパッケージ化に Extras メカニズムを使用していないため、現時点では、core リポジトリと livepatch リポジトリの 2 つの RPM リポジトリしかなく、それぞれにアドバイザリが含まれています。livepatch レポジトリは、[AL2023 のカーネルライブパッチ](#)用です。

アドバイザリのリリース日とアドバイザリの更新日

Amazon Linux Advisories のアドバイザリリリース日は、セキュリティ更新が RPM リポジトリで最初に公開された日時を示します。アドバイザリは、RPM リポジトリを介して修正プログラムがインストール可能になった直後に、[Amazon Linux Security Center](#) ウェブサイトに掲載されます。

アドバイザリの更新日は、以前に公開されたアドバイザリに新しい情報が追加された日付を示します。

AL2023 のバージョン番号 (2023.6.20241031 など) と、そのリリースと同時に公開されたアドバイザリのアドバイザリリリース日の間には、なんらの関連性をも想定すべきではありません。

アドバイザリのタイプ

RPM リポジトリのメタデータは、さまざまなタイプのアドバイザリをサポートしています。Amazon Linux はほぼ例外なく、セキュリティ更新に関するアドバイザリのみを発表していますが、これに限ると想定すべきではありません。バグ修正、機能強化、新しいパッケージなどのイベントに関するアドバイザリが発表され、アドバイザリにそのようなタイプの更新が含まれているとマークされる可能性があります。

アドバイザリの重大度

各アドバイザリでは、各問題は個別に評価されるため、独自の重大度が指定されます。単一のアドバイザリで複数の CVE について対応する場合もあり、各 CVE の評価はさまざまでも、アドバイザリ自体の重大度は 1 つです。単一のパッケージ更新についてのアドバイザリが複数存在する場合もあります。このため、特定のパッケージ更新に、複数の重大度 (アドバイザリごとに 1 つ) が存在する場合があります。

Amazon Linux では、アドバイザリの重大度を示すのに、重大度の高い順に、Critical (重大)、Important (重要)、Moderate (中)、Low (低) を使用しています。Amazon Linux アドバイザリには重大度が指定されていない場合もありますが、これは非常にまれです。

Amazon Linux は、Moderate (中) という用語を使用する RPM ベースの Linux ディストリビューションの 1 つであり、他の RPM ベースの Linux ディストリビューションでは、同等の用語である Medium (中) を使用しています。Amazon Linux パッケージマネージャーは両方の用語を同義として扱いますが、サードパーティーのパッケージリポジトリでは Medium の用語を使用する場合があります。

Amazon Linux Advisories では、対処されている関連する問題について経時的に新たな知見が得られるにつれ、随時重大度が変更される可能性があります。

通常、アドバイザリの重大度は、アドバイザリで参照される CVE について、Amazon Linux で評価された CVSS スコアの最大値に基づいて産出されます。ただし、必ずしもそうとは限りません。例えば、CVE が割り当てられていない問題に対処している場合などがこれにあたります。

Amazon Linux がアドバイザリの重大度評価を使用する方法の詳細については、「[ALAS FAQ](#)」を参照してください。

アドバイザリとパッケージ

単一のパッケージに対して複数のアドバイザリが発表される場合がありますが、すべてのパッケージに対してアドバイザリが発表されるとは限りません。特定のパッケージのバージョンは、複数のアドバイザリで参照される可能性があり、各アドバイザリに独自の重大度と CVE が指定されます。

同じパッケージ更新に対して複数のアドバイザリが 1 つの新しい AL2023 リリースで同時に発表される場合も、立て続けに発表される場合もあります。

他の Linux ディストリビューションと同様に、同じソースパッケージから単一または多数の異なるバイナリパッケージが構築される場合があります。例えば、[ALAS-2024-698](#) は、mariadb105 パッケージに適用されるものとして、[Amazon Linux Security Center ウェブサイトの AL2023 セクション](#)に掲載されているアドバイザリです。これはソースパッケージ名であり、アドバイザリ自体はソースパッケージとともにバイナリパッケージも参照しています。この場合、1 つの mariadb105 ソースパッケージから 12 を超えるバイナリパッケージが構築されています。ソースパッケージと同じ名前のバイナリパッケージが存在することは非常によくあることですが、必ずしもそうであるとは限りません。

Amazon Linux Advisories は通常、更新されたソースパッケージから構築されたすべてのバイナリパッケージを一覧表示していますが、必ずそうであると想定すべきではありません。パッケージマネージャーと RPM リポジトリのメタデータ形式により、更新されたバイナリパッケージのサブセットを一覧表示するアドバイザリが出される場合もあります。

特定のアドバイザリが、特定の CPU アーキテクチャにのみ適用される場合もあります。すべてのアーキテクチャ用に構築されていないパッケージや、すべてのアーキテクチャに影響を与えない問題がある場合もあります。パッケージがすべてのアーキテクチャで利用可能であるのに、問題が 1 つのアーキテクチャにのみ適用される場合、Amazon Linux は通常、影響を受けるアーキテクチャのみを参照するアドバイザリが発表されることはありませんが、その可能性を排除すべきではありません。

パッケージの依存関係の特質上、アドバイザリが 1 つのパッケージを参照していても、その更新をインストールするには、アドバイザリに記載されていないパッケージを含むその他のパッケージの更新が必要になる場合がよくあります。dnf パッケージマネージャーは、必要な依存関係のインストールを処理します。

アドバイザリと CVE

アドバイザリは、0 以上の CVE に対応している場合があります、同じ CVE を参照するアドバイザリが複数ある場合もあります。

アドバイザリが 0 件の CVE を参照する例には、問題に CVE がまだ割り当てられていない場合 (またはまったく割り当てられない場合) があります。

複数のアドバイザリが同じ CVE を参照する例には、CVE が複数のパッケージに適用される場合などがあります。例えば、[CVE-2024-21208](#) は、Corretto 8、11、17、21 に適用されます。これらの Corretto バージョンは、それぞれ AL2023 の個別のパッケージであり、以下のとおり、各パッケージに対してアドバイザリが出されています。Corretto 8 には [ALAS-2024-754](#)、Corretto 11 には [ALAS-2024-753](#)、Corretto 17 には [ALAS-2024-752](#)、Corretto 21 には [ALAS-2024-752](#) です。これらの Corretto リリースはすべて同じ CVE のリストがありますが、必ずしもそうであるとは限らないことに注意が必要です。

特定の CVE が、パッケージごとに異なる評価を受ける場合があります。例えば、重要度が重要であるアドバイザリで特定の CVE が参照されている場合に、同じ CVE について異なる重大度で参照する別のアドバイザリが出されている場合もあります。

RPM リポジトリメタデータでは、各アドバイザリのリファレンスを一覧表示できます。Amazon Linux では通常、CVE のみが参照されますが、メタデータ形式では他のリファレンスタップも使用できます。

RPM パッケージリポジトリのメタデータは、修正可能な CVE のみを参照します。[Amazon Linux Security Center ウェブサイトの「Explore」セクション](#)には、Amazon Linux が評価した CVE に関する情報が掲載されています。この評価により、さまざまな Amazon Linux リリースおよびパッケージの CVSS ベーススコア、重大度、ステータスが算出される可能性があります。特定の Amazon Linux リリースまたはパッケージの CVE のステータスは、Not Affected (影響なし)、Pending Fix (修正保留中)、または No Fix Planned (修正予定なし) のいずれかになります。CVE のステータスと評価は、アドバイザリが発表される前に何度も任意の方法で変更される可能性があります。これには、Amazon Linux への CVE の適用可能性が再評価された場合などがあります。

アドバイザリで参照される CVE のリストは、そのアドバイザリの最初の公開後に変更される可能性があります。

アドバイザリのテキスト

アドバイザリには、アドバイザリが作成される理由となった問題を説明するテキストも含まれます。通常、このテキストは、未修正の CVE のテキストです。このテキストでは、Amazon Linux が修正

を適用したパッケージバージョンとは異なる、修正が利用可能なアップストリームバージョン番号を参照している場合があります。新しいアップストリームリリースから、Amazon Linux が修正をバックポートすることはよくあります。アドバイザリのテキストに、Amazon Linux バージョンで出荷されたバージョンとは異なるアップストリームリリースが記載されている場合でも、アドバイザリ内の Amazon Linux パッケージのバージョンは正確です。

RPM リポジトリメタデータ内のアドバイザリテキストは、詳細についてはシンプルに [Amazon Linux Security Center](#) ウェブサイトを参照するプレースホルダーテキストである場合もあります。

カーネルライブパッチのアドバイザリ

ライブパッチに関するアドバイザリは、アドバイザリの対象となるパッケージ (kernel-livepatch-6.1.15-28.43 など) とは異なるパッケージ (Linux カーネル) を参照するという点で独特です。

[カーネルライブパッチ](#)に関するアドバイザリでは、ライブパッチパッケージが適用される特定のカーネルバージョンについて、そのライブパッチパッケージが対処できる問題 (CVE など) が参照されます。

各ライブパッチは、特定のカーネルバージョンを対象としています。CVE にライブパッチを適用するには、使用するカーネルバージョンに適したライブパッチパッケージをインストールし、ライブパッチを適用する必要があります。

例えば、[CVE-2023-6111](#) は、AL2023 カーネルバージョン

6.1.56-82.125、6.1.59-84.139、6.1.61-85.141 に対してライブパッチを適用できます。この CVE を修正した新しいカーネルバージョンもリリースされており、[別のアドバイザリ](#)が出されています。AL2023 で [CVE-2023-6111](#) に対処するには、[ALAS2023-2023-461](#) で指定されているバージョン移行のカーネルバージョンを実行するか、この CVE に対するライブパッチが適用されたカーネルバージョンのいずれかを実行する必要があります。

既にライブパッチが利用可能な特定のカーネルバージョンに新しいライブパッチが利用可能になった場合、kernel-livepatch-KERNEL_VERSION パッケージの新しいバージョンがリリースされます。例えば、[ALASLIVEPATCH-2023-003](#) アドバイザリは、3 つの CVE に対応する 6.1.15-28.43 カーネルのライブパッチを含む kernel-livepatch-6.1.15-28.43-1.0-1.amzn2023 パッケージで公開されました。その後、[ALASLIVEPATCH-2023-009](#) アドバイザリが kernel-livepatch-6.1.15-28.43-1.0-2.amzn2023 パッケージとともに公開されました。これは、別の CVE に対するライブパッチを含む、6.1.15-28.43 カーネルの以前のライブパッチパッケージの更新でした。他のカーネルバージョンにもライブパッチアドバイザリの問題があり、パッケージにはそれらの特定のカーネルバージョンを対象とするライブパッチも含まれていました。

カーネルライブパッチの詳細については、「[AL2023 のカーネルライブパッチ](#)」を参照してください。

セキュリティアドバイザリに関連するツールを開発している場合は、「[アドバイザリと updateinfo.xml の XML スキーマ](#)」セクションを参照して詳細を確認することをお勧めします。

アドバイザリと updateinfo.xml の XML スキーマ

updateinfo.xml ファイルは、パッケージリポジトリ形式の一部です。これは、[適用可能なアドバイザリの一覧表示](#)や[セキュリティ更新プログラムのインプレースでの適用](#)などの機能を実装するために dnf パッケージマネージャーが解析するメタデータです。

リポジトリのメタデータ形式を解析するためにカスタムコードを作成するのではなく、dnf パッケージマネージャーの API を使用することをお勧めします。AL2023 の dnf バージョンは、AL2023 リポジトリ形式と AL2 リポジトリ形式の両方を解析できるため、どちらの OS バージョンのアドバイザリ情報も、API を使用して調べることができます。

[RPM Software Management](#) プロジェクトは、GitHub の [rpm-metadata](#) リポジトリに RPM メタデータ形式をドキュメント化しています。

updateinfo.xml メタデータを直接解析するツールを開発している場合は、[rpm-metadata ドキュメント](#)をよく読むことを強くお勧めします。このドキュメントでは、実際に使用されている事例を取り上げており、メタデータ形式のルールとして合理的に解釈できるものに対する多くの例外が網羅されています。

GitHub の [raw-historical-rpm-repository-examples](#) リポジトリには、updateinfo.xml ファイルの実例の例が数多く掲載されており、現在も増え続けています。

ドキュメントで不明な点がある場合は、GitHub プロジェクトで Issue を開いてお問い合わせください。回答してドキュメントを適切に更新します。これはオープンソースプロジェクトであるため、ドキュメントの更新に関するプルリクエストも歓迎いたします。

適用可能なアドバイザリの一覧表示

dnf パッケージマネージャーは、どのアドバイザリがどのパッケージバージョンで修正されたかを示すメタデータにアクセスできます。これにより、インスタンスまたはコンテナイメージに適用可能なアドバイザリを一覧表示できます。

Note

[AWS Systems Manager](#) などのツールでは、この機能を使用して、単一のインスタンスだけでなく、フリート全体にわたって関連する更新を表示できます。

更新を一覧表示する場合、特定の AL2023 リリースのメタデータ、または最新リリースのメタデータを確認するように `dnf` に指示できます。

Note

AL2023 リリースが行われると、リリースはイミュータブルになります。したがって、[Amazon Linux セキュリティセンター](#)での新規または更新されたアドバイザリは、AL2023 の新しいリリースのメタデータにのみ追加されます。

どのアドバイザリを AL2023 コンテナイメージに適用するかのをいくつか以下に示します。これらのコマンドはすべて、EC2 インスタンスなどのコンテナ化されていない環境で機能します。

Listing advisories in a specific version

この例では、[2023.1.20230628](#) リリースのどのアドバイザリが [2023.0.20230315](#) リリースのコンテナイメージに関連しているかを確認します。

Note

この例では、[2023.0.20230315](#) および [2023.1.20230628](#) リリースを使用しています。これらは AL2023 の最新リリースではありません。最新のセキュリティ更新プログラムを含む最新リリースについては、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.0.20230315](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の `.0` は、特定のリリースにおけるイメージのバージョンを示します。このイメージのバージョンは通常 0 です。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこからコンテナにインストールされているパッケージに関連するアドバイザリを一覧表示するよう `dnf` に指示します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

次に `dnf updateinfo` コマンドを使用して、[2023.1.20230628](#) リリースのどのアドバイザリが、インストール済みのパッケージに関連しているかの概要を表示します。

```
$ 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)
```

アドバイザリのリストを取得するには、`--list` オプションを `dnf updateinfo` に指定できます。

```
$ 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

この例では、[2023.4.20240319](#) リリースのコンテナを起動した場合、どの更新が AL2023 の latest バージョンで利用可能であるかを確認します。執筆時点で、latest リリースは [2023.5.20240708](#) であるため、この例に記載されている更新は、そのリリース時点のものです。

 Note

この例では [2023.4.20240319](#) リリースと [2023.5.20240708](#) リリースを使用しており、後者は執筆時点での最新リリースです。最新リリースの詳細については、「[AL2023 リリースノート](#)」を参照してください。

この例では、[2023.4.20240319](#) リリースのコンテナイメージから始めます。

まず、このコンテナイメージをコンテナレジストリから取得します。末尾の .1 は、特定のリリースにおけるイメージのバージョンを示します。イメージバージョンは通常ゼロですが、この例ではイメージバージョンが 1 のリリースを使用します。

```
$ 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
```

これでコンテナ内にシェルを立ち上げられるようになったので、そこから更新を確認します。

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

次に `dnf updateinfo` コマンドを使用して、インストールされたパッケージに関連する最新リリースのアドバイザリの概要を表示します。執筆時点では、[2023.1.20230628](#) が最新リリースでした。

```
$ 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)
```

アドバイザリの一覧を取得するには、`--list` オプションを `dnf updateinfo` に指定できます。

```
$ 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
```

セキュリティ更新プログラムのインプレースでの適用

更新プログラムの適用の概要については、「[DNF およびリポジトリバージョンを使用してセキュリティ更新を適用します。](#)」を参照してください。 `dnf upgrade` で `--security` オプションを指定すると、パッケージ更新プログラムが、アドバイザリに記載されているもののみに制限されます。このセクションの残りの部分では、特定のセキュリティ更新プログラムのみをインストールする方法について説明します。

Note

新しい AL2023 リリースで利用可能なすべての更新を適用することをお勧めします。セキュリティ更新プログラムのみ、または特定の更新プログラムのみを選択するのは例外とし、ルールにはしないでください。

アドバイザリに記載されている更新プログラムの適用

`dnf upgradeinfo` の出力の先頭列のアドバイザリ識別子を使用して、アドバイザリに記載されているパッケージの更新プログラムを適用できます。アドバイザリに記載されたパッケージを利用可能な最新のパッケージに更新するか、アドバイザリに記載されているバージョンまでしか更新しないかを、`dnf` パッケージマネージャーで指示できます。更新プログラムが既にインストールされている場合、更新コマンドは `no-op` となります。

対象のパッケージに対してアドバイザリに記載されているバージョンまで更新を適用するには、`dnf upgrade-minimal` コマンドを使用し、`--advisory` オプションでアドバイザリを指定します。次の例は、AL2023 バージョン [2023.0.20230315](#) コンテナで `dnf upgrade-minimal` を実行しています。

```
$ 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!

この要求は、アドバイザリに対処するために必要な最小限の更新を `dnf` に実行させるものであるため、`--releasever=latest` が使用された場合でも同じパッケージバージョンが更新されます。

通常の `dnf upgrade` コマンドを `--advisory` オプションとともに使用すると、アドバイザリに記載されている該当パッケージが、利用可能な最新のバージョンに更新されます。このバージョンは、アドバイザリに記載されているバージョンよりも新しい場合があります。

Note

`system-release` パッケージが更新されない限り、`dnf` が固定されている AL2023 リポジトリのバージョンは変更されません。

Warning

`dnf` が固定されているリポジトリのバージョンを変更せずに AL2023 の別のリリースから更新プログラムをインストールする場合は、以降の `dnf` の変更操作において注意が必要になります。例えば、パッケージをインストールまたは更新する際、新しいリリースではパッケージの依存関係が変更されている可能性があるため、以前のリリースのままではこれらの新しい依存関係に対応できない場合があります。

次の例は、AL2023 [バージョン 2023.0.20230315](#) コンテナで、AL2023 の最新リリース (執筆時点では [2023.5.20240708](#)) を参照して実行されます。更新後の `curl` バージョンがいずれも `update-minimal` で更新されるバージョンよりも新しいものであると同時に、この新しいバージョンでは新しい依存関係が追加されている点に注意してください。

```
$ 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!

AL2023 の SELinux モードの設定

AL2023 のデフォルトでは、Security Enhanced Linux (SELinux) は enabled であり、permissive モードに設定されています。許可モードでは、アクセス拒否は記録されますが、強制ではありません。SELinux は、カーネルの主要なサブシステムに強力な柔軟性のある強制アクセス制御 (MAC) アーキテクチャを提供するためのカーネル機能とユーティリティの集まりです。

SELinux は、機密性と完全性の要件に基づいて情報を分離するように強化されたメカニズムを備えています。このように情報を分離することで、アプリケーションのセキュリティメカニズムの改ざんやバイパスの脅威が軽減されます。また、悪意のあるアプリケーションや欠陥のあるアプリケーションによって引き起こされる可能性のあるダメージも回避できます。

SELinux には、日々のセキュリティの目標を満たすように設計されたサンプルセキュリティポリシー設定ファイルのセットが含まれています。

SELinux の特徴と機能の詳細については、「[SELinux ノートブック](#)」および「[ポリシー言語](#)」を参照してください。

トピック

- [AL2023 でのデフォルトの SELinux のステータスとモード](#)
- [enforcing モードへの変更](#)

- [AL2023 の SELinux を無効にするオプション](#)

AL2023 でのデフォルトの SELinux のステータスとモード

AL2023 では、SELinux はデフォルトで **enabled** となり、**permissive** モードに設定されています。**permissive** モードでは、アクセス拒否は記録されますが、強制ではありません。

getenforce または **sestatus** コマンドは、現在の SELinux のステータス、ポリシー、モードを知ることができます。

デフォルトステータスが **enabled** およびに **permissive** 設定されていると、**getenforce** コマンドは **permissive** を返します。

以下の例で示すように、**sestatus** コマンドは SELinux のステータスと現在の SELinux ポリシーを返します。

```
$ 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
```

SELinux を **permissive** モードで実行すると、ユーザーがファイルに誤ったラベルを付ける可能性があります。SELinux を **disabled** ステータスで実行すると、ファイルにはラベルが付けられません。**enforcing** モードに変更すると、正しくないファイルやラベル付けされていないファイルの両方が問題を引き起こす可能性があります。

SELinux は、この問題を回避するために自動的にファイルにラベルを付け直します。SELinux は、ステータスを **enabled** に変更したときの自動再ラベル付けにより、ラベル付けの問題を防ぎます。

enforcing モードへの変更

SELinux を **enforcing** モードで実行すると、SELinux ユーティリティによって設定済みのポリシーが **enforcing** されます。SELinux は、ポリシーのルールに基づいてアクセスを許可または拒否することで、特定のアプリケーションの機能を管理します。

SELinux の現在のモードを調べるには、`getenforce` コマンドを実行します。

```
getenforce
Permissive
```

設定ファイルを編集して **enforcing** モードを有効にする

モードを `enforcing` に変更するには、次の手順を使用します。

1. `/etc/selinux/config` ファイルを編集して `enforcing` モードに変更します。SELINUX 設定は次の例のようになります。

```
SELINUX=enforcing
```

2. システムを再起動して `enforcing` モードへの変更を完了します。

```
$ sudo reboot
```

次の起動時に、SELinux はシステム内のすべてのファイルとディレクトリにラベルを付け直します。SELinux は、SELinux が `disabled` であったときに作成されたファイルとディレクトリの SELinux コンテキストも追加します。

`enforcing` モードに変更すると、SELinux ポリシールールが不正確または欠落しているという理由で SELinux が一部のアクションを拒否することがあります。以下のコマンドを使用して SELinux が拒否するアクションを表示できます。

```
$ sudo ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

cloud-init を使用して **enforcing** モードを有効にする

別の方法として、インスタンスを起動するときに、以下の `cloud-config` をユーザーデータとして渡して、`enforcing` モードを有効にします。

```
#cloud-config
selinux:
  mode: enforcing
```


デフォルトでは、この設定によりインスタンスが再起動されます。安定性を高めるため、インスタンスを再起動することをお勧めします。ただし、必要に応じて、いかなる cloud-config を指定して再起動をスキップすることもできます。

```
#cloud-config
selinux:
  mode: enforcing
  selinux_no_reboot: 1
```

AL2023 の SELinux を無効にするオプション

SELinux を無効にすると、SELinux ポリシーが読み込まれたり適用されたりせず、Access Vector Cache (AVC) メッセージはログに記録されません。SELinux を実行することで得られる利点はすべて失われます。

SELinux を無効にする代わりに、permissive モードを使用することをお勧めします。SELinux を完全に無効にするよりも、permissive モードで実行するほうが少しだけコストが高くなります。permissive モードから enforcing モードへの移行するほうが、SELinux を無効にした後に enforcing モードに戻るよりも設定の調整がはるかに少なく済みます。ファイルにラベルを付けることができ、アクティブなポリシーで拒否された可能性のあるアクションをシステムが追跡して記録できます。

SELinux から **permissive** モードへの変更

SELinux を permissive モードで実行すると、SELinux ポリシーは適用されません。permissive モードでは、SELinux は AVC メッセージをログに記録しますが、操作は拒否されません。これらの AVC メッセージは、トラブルシューティング、デバッグ、SELinux ポリシーの改善に使用できます。

SELinux を許可モードに変更するには、次の手順を使用します。

1. /etc/selinux/config ファイルを編集して permissive モードに変更します。SELINUX 値は次の例のようになります。

```
SELINUX=permissive
```

2. システムを再起動して permissive モードへの変更を完了します。

```
sudo reboot
```

SELinux の無効化

SELinux を無効にすると、SELinux ポリシーが読み込まれたり適用されたりせず、AVC メッセージはログに記録されません。SELinux を実行することで得られる利点はすべて失われます。

SELinux を無効にするには次の手順を使用します。

1. grubby パッケージがインストールされていることを確認します。

```
rpm -q grubby  
grubby-version
```

2. ブートローダーをカーネルコマンドラインに `selinux=0` を追加するように設定します。

```
sudo grubby --update-kernel ALL --args selinux=0
```

3. システムを再起動します。

```
sudo reboot
```

4. `getenforce` コマンドを実行して SELinux が Disabled であることを確認します。

```
$ getenforce  
Disabled
```

SELinux について詳しくは、[SELinux ノートブック](#)および [SELinux の設定](#)を参照してください。

AL2023 で FIPS モードを有効にする

このセクションでは、AL2023 で連邦情報処理標準 (FIPS) を有効にする手順について説明します。FIPS の詳細については、以下を参照してください。

- [連邦情報処理標準 \(FIPS\)](#)
- [コンプライアンスのよくある質問: 連邦情報処理標準](#)

Note

このセクションでは、AL2023 で FIPS モードを有効にする方法について説明します。AL2023 暗号モジュールの認証状況についての説明はありません。

前提条件

- インターネットにアクセスして必要なパッケージをダウンロードできる既存の AL2023 (AL2023.2 以上) Amazon EC2 インスタンス。AL2023 Amazon EC2 インスタンスの起動の詳細については、「[Amazon EC2 コンソールを使用した AL2023 の起動](#)」を参照してください。
- SSH または AWS Systems Manager を使用して Amazon EC2 インスタンスに接続する必要があります。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。

Important

ED25519 SSH ユーザーキーは FIPS モードではサポートされていません。ED25519 SSH key pair を使用して Amazon EC2 インスタンスを起動した場合、別のアルゴリズム (RSA など) を使用して新しいキーを生成する必要があります。そうでない場合は、FIPS モードを有効にした後にインスタンスにアクセスできなくなる可能性があります。詳細については、「Amazon EC2 ユーザーガイド」の「[キーペアを作成する](#)」を参照してください。

FIPS モードの有効化

- SSH または AWS Systems Manager を使用して AL2023 インスタンスに接続します。
- システムが最新であることを確認します。詳細については、「[AL2023 のパッケージとオペレーティングシステムとの更新を管理する](#)」を参照してください。
- crypto-policies ユーティリティがインストールされていて、最新であることを確認します。

```
sudo dnf -y install crypto-policies crypto-policies-scripts
```

- 以下のコマンドを実行して、FIPS モードを有効にします。これにより、「[Amazon Linux 2023 に関するよくある質問](#)」に記載されているモジュールに対して、システム全体で FIPS モードが有効になります。

```
sudo fips-mode-setup --enable
```

5. 以下のコマンドを実行して、インスタンスを再起動します。

```
sudo reboot
```

6. FIPS モードが有効であることを確認するには、インスタンスに再接続し、以下のコマンドを実行します。

```
sudo fips-mode-setup --check
```

以下の出力例は、FIPS モードが有効であることを示しています。

```
FIPS mode is enabled.
```

AL2023 コンテナで FIPS モードを有効にする

このセクションでは、AL2023 コンテナで連邦情報処理標準 (FIPS) を有効にする手順を説明します。FIPS の詳細については、以下を参照してください。

- [連邦情報処理標準 \(FIPS\)](#)
- [コンプライアンスのよくある質問: 連邦情報処理標準](#)

Note

このセクションでは、AL2023 コンテナで FIPS モードを有効にする方法を説明します。AL2023 暗号化モジュールの認証状況については取り上げません。

前提条件

- インターネットにアクセスして必要なパッケージをダウンロードできる既存の AL2023 (AL2023.2 以上) Amazon EC2 インスタンス。AL2023 Amazon EC2 インスタンスの起動の詳細については、「[Amazon EC2 コンソールを使用した AL2023 の起動](#)」を参照してください。
- SSH または AWS Systems Manager を使用して Amazon EC2 インスタンスに接続する必要があります。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。

⚠ Important

`fips-mode-setup` コマンドはコンテナ内からは正しく動作しません。AL2023 コンテナで FIPS モードを適切に設定するには、以下の手順を参照してください。

AL2023 コンテナで FIPS モードを有効にする

- 最初に AL2023 コンテナのホストで FIPS モードを有効にする必要があります。「[AL2023 で FIPS モードを有効にする](#)」の手順に従って、ホストで FIPS モードを有効にします。
- SSH または AWS Systems Manager を使用して AL2023 コンテナのホストインスタンスに接続します。
- AL2023 ホストが FIPS モードになり、コンテナ内から `/proc/sys/crypto/fips_enabled` へのアクセスが可能な場合、AL2023 コンテナで FIPS モードが自動的に有効になります。`/proc/sys/crypto/fips_enabled` の値が 0 の場合は FIPS が有効ではなく、値が 1 の場合は FIPS モードが有効になっていることを示します。

AL2023 のホストとコンテナの両方で、次のコマンドを実行して FIPS が有効になっていることを確認できます。

```
cat /proc/sys/crypto/fips_enabled
```

- 次に、コンテナ内で FIPS 暗号化ポリシーを有効にします。これを行うには、以下のオプションで説明されているような方法があります。環境に応じて最適なオプションを使用してください。
 - `update-crypto-policies` コマンドを使用して、コンテナ内で FIPS 暗号化ポリシーを手動で有効にします。

```
# Run these commands inside the container
dnf install -y crypto-policies-scripts
update-crypto-policies --set FIPS
```

- AL2023 コンテナ内に `bind` マウントを作成します (これは他のディストリビューションでの `podman` の動作と似ています)。

```
# 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. AL2023 コンテナが AL2023 ホストの暗号化ポリシーと一致するようにバインドマウントを作成することもできます。以下はあくまで例として示しています。この設定では、コンテナとホストの間で暗号化ポリシーとパッケージのバージョンに互換性のない差異がある場合に問題が発生する可能性があります。

```
sudo docker pull amazonlinux:2023
sudo docker run --mount type=bind,readonly,src=/etc/crypto-policies,dst=/etc/
crypto-policies -it amazonlinux:2023
```

5. 上記の手順を実行したら、次のコマンドを使用して、コンテナで FIPS が有効になっていることを再度確認できます。

```
$ cat /etc/crypto-policies/config
FIPS

$ cat /proc/sys/crypto/fips_enabled
1
```

AL2023 で OpenSSL FIPS プロバイダーを切り替える

このセクションでは、AL2023 で latest と certified の OpenSSL FIPS プロバイダーを切り替える方法について説明します。

FIPS の詳細については、以下を参照してください。

- [連邦情報処理標準 \(FIPS\)](#)
- [コンプライアンスのよくある質問: 連邦情報処理標準](#)
- [暗号化モジュールの選択と使用のための FedRAMP ポリシー](#)

Important

AL2023.7 以降では、デフォルトの OpenSSL FIPS プロバイダーは openssl-fips-provider-latest パッケージであり、定期的にバグ修正とセキュリティ更新プログラムを受け取ります。

以下の手順は、`openssl-fips-provider-certified` パッケージに固定することを希望するお客様を対象としています。このバージョンの FIPS プロバイダーは、NIST 証明書のチェックサムと一致し、最新の更新プログラムがない可能性があります。FIPS 認定モジュールとパッケージバージョンの詳細については、「[Amazon Linux 2023 に関するよくある質問](#)」を参照してください。

前提条件

- インターネットにアクセスして必要なパッケージをダウンロードできる既存の AL2023 (AL2023.7 以上) Amazon EC2 インスタンス。AL2023 Amazon EC2 インスタンスの起動の詳細については、「[Amazon EC2 コンソールを使用した AL2023 の起動](#)」を参照してください。
- SSH または AWS Systems Manager を使用して Amazon EC2 インスタンスに接続する必要があります。詳細については、「[AL2023 インスタンスへの接続](#)」を参照してください。
- AL2023 で FIPS モードを有効にするには、「[AL2023 で FIPS モードを有効にする](#)」の手順に従います。

`openssl-fips-provider-latest` と `openssl-fips-provider-certified` の間で切り替える

1. `dnf` を使用して OpenSSL FIPS プロバイダーを切り替えます。

```
sudo dnf -y swap openssl-fips-provider-latest openssl-fips-provider-certified
```

2. 認定された OpenSSL FIPS プロバイダーを使用していることを確認します。FIPS モードの AL2023 で、次のコマンドを実行します。

```
openssl list -providers
```

以下の出力が表示されます。

```
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 カーネル強化

AL2023 の 6.1 Linux カーネルは、いくつかの強化オプションと機能を使用して設定および構築されています。

カーネル強化オプション (アーキテクチャに依存していません)

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_ACPI_CUSTOM_METHOD	n	n	該当なし	該当なし	該当なし	該当なし
CONFIG_BINFMT_MISC	m	m	m	m	m	m
CONFIG_BUG	y	y	y	y	y	y
CONFIG_BUG_ON_DATA_CORRUPTION	y	y	y	y	y	y
CONFIG_CLANG	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_CFI_PERMISSIVE	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_COMPAT	y	y	y	y	y	y
CONFIG_COMPAT_BRK	n	n	n	n	n	n
CONFIG_COMPAT_VDSO	該当なし	n	該当なし	n	該当なし	n
CONFIG_DEBUG_CREDENTIALS	n	n	該当なし	該当なし	該当なし	該当なし
CONFIG_DEBUG_LIST	y	y	y	y	y	y
CONFIG_DEBUG_NOTIFIERS	n	n	n	n	n	n
CONFIG_DEBUG_SG	n	n	n	n	n	n
CONFIG_DEBUG_VIRTUAL	n	n	n	n	n	n
CONFIG_DEBUG_WX	n	n	n	n	n	n

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_DEFALT_MMAP_MIN_ADDR	65536	65536	65536	65536	65536	65536
CONFIG_DEFAULT_VMEMMAP	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_DEFAULT_VMEMMAP	n	n	n	n	n	n
CONFIG_DEFAULT_I_DISABLE_PCI_DMA	n	n	n	n	n	n
CONFIG_DEFAULT_RTIFY_SOURCE	y	y	y	y	y	y
CONFIG_HARDENED_USERCOPY	y	y	y	y	y	y
CONFIG_HARDENED_USERCOPY_FALLBACK	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_HARDENED_USERCOPY_PAGESPAN	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
<u>CONFIG_HIBERNATION</u>	y	y	y	y	y	y
<u>CONFIG_HW_RANDOM_TPM</u>	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
<u>CONFIG_INET_DIAG</u>	m	m	m	m	m	m
<u>CONFIG_INET_ON_ALL_OC_DEFAULT_T_ON</u>	n	n	n	n	n	n
<u>CONFIG_INET_ON_FREE_DEFAULT_ON</u>	n	n	n	n	n	n
<u>CONFIG_INET_STACK_ALL_ZERO</u>	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
<u>CONFIG_IOMMU_DEFAULT_DMA_STRICT</u>	n	n	n	n	n	n
<u>CONFIG_IOMMU_SUPPORT</u>	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
<u>CONFIG_IO_STRICT_DEVMEM</u>	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
<u>CONFIG_KEXEC</u>	y	y	y	y	y	y
<u>CONFIG_KFENCE</u>	n	n	n	n	n	n
<u>CONFIG_LOAD_UNLOAD</u>	n	n	n	n	n	n
<u>CONFIG_LEGACY_PTYS</u>	n	n	n	n	n	n
<u>CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY</u>	n	n	n	n	n	n
<u>CONFIG_MODULES</u>	y	y	y	y	y	y
<u>CONFIG_MODULE_SIG</u>	y	y	y	y	y	y
<u>CONFIG_MODULE_SIG_ALL</u>	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
<u>CONFIG_MODULE_SIG_FORCE</u>	n	n	n	n	n	n
<u>CONFIG_MODULE_SIG_HASH</u>	sha512	sha512	sha512	sha512	sha512	sha512
<u>CONFIG_MODULE_SIG_KEY</u>	certs/sig ning_key. pem	certs/sig ning_key. pem	certs/sig ning_key. pem	certs/sig ning_key. pem	certs/sig ning_key. pem	certs/sig ning_key. pem
<u>CONFIG_MODULE_SIG_SHA512</u>	y	y	y	y	y	y
<u>CONFIG_PAGE_POISONING</u>	n	n	n	n	n	n
<u>CONFIG_PAGE_POISONING_NO_SANITY</u>	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
<u>CONFIG_PAGE_POISONING_ZERO</u>	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
<u>CONFIG_PANIC_ON_OOPS</u>	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_PANIC_TIMEOUT	0	0	0	0	0	0
CONFIG_PROC_KCORE	y	y	y	y	y	y
CONFIG_RANDOMIZE_KERNEL_STACK_OFFSET_DEFAULT	n	n	n	n	n	n
CONFIG_RANDOMIZE_KERNEL_TRUST_BOOTLOADER	y	y	該当なし	該当なし	該当なし	該当なし
CONFIG_RANDOMIZE_KERNEL_TRUST_CPU	y	y	該当なし	該当なし	該当なし	該当なし
CONFIG_REFCOUNT_FULL	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_SCHED_CORE	該当なし	y	該当なし	y	該当なし	y
CONFIG_SCHED_STACK_END_CHECK	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
<u>CONFIG_SECCOMP</u>	y	y	y	y	y	y
<u>CONFIG_SECCOMP_FILTER</u>	y	y	y	y	y	y
<u>CONFIG_SECURITY</u>	y	y	y	y	y	y
<u>CONFIG_SECURITY_DMESG_RESTRICT</u>	y	y	y	y	y	y
<u>CONFIG_SECURITYLANDLOCK</u>	y	y	y	y	y	y
<u>CONFIG_SECURITY_LOADCKDOWN_LSM</u>	y	y	y	y	y	y
<u>CONFIG_SECURITY_LOADCKDOWN_LSM_EARLY</u>	y	y	y	y	y	y
<u>CONFIG_SECURITY_LinuxBootParams</u>	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_SECURITY_SELINUX_DISABLE	y	y	y	y	y	y
CONFIG_SECURITY_SELINUX_DISABLE	n	n	該当なし	該当なし	該当なし	該当なし
CONFIG_SECURITY_WRITABLE_HOOKS	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_SECURITY_YAMA	y	y	y	y	y	y
CONFIG_SHUFFLE_PAGE_ALLOCATOR	y	y	y	y	y	y
CONFIG_SLAB_FREELIST_HARDENED	y	y	y	y	y	y
CONFIG_SLAB_FREELIST_RANDOM	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_SLUB_DEBUG	y	y	y	y	y	y
CONFIG_STACKPROTECTOR	y	y	y	y	y	y
CONFIG_STACKPROTECTOR_STRONG	y	y	y	y	y	y
CONFIG_STATIC_USERMODEHELPER	n	n	n	n	n	n
CONFIG_STRICT_DEVMEM	n	n	n	n	n	n
CONFIG_STRICT_KERNEL_RWX	y	y	y	y	y	y
CONFIG_STRICT_MODULE_RWX	y	y	y	y	y	y
CONFIG_SYSCALL_COOKIES	y	y	y	y	y	y
CONFIG_VMAP_STACK	y	y	y	y	y	y

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_WERROR	n	n	n	n	n	n
CONFIG_ZERO_CALL_USED_REGS	n	n	n	n	n	n

ACPI メソッドをランタイムに挿入/置換できるようにする (CONFIG_ACPI_CUSTOM_METHOD)

Amazon Linux では、root ユーザーが任意のカーネルメモリに書き込むことができるため、このオプションは無効になっています。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

その他のバイナリフォーマット (binfmt_misc)

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。AL2023 では、この機能はオプションで、カーネルモジュールとして構築されています。

BUG() サポート

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

カーネルメモリ構造の有効性をチェックしているときに、カーネルがデータ破損に遭遇した場合の **BUG()**

Linux カーネルの一部は、データ構造の内部整合性をチェックし、データ破損を検出した場合は **BUG()** できます。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

COMPAT_BRK

このオプションを無効にすると (Amazon Linux がカーネルを設定する方法)、`randomize_va_space` `sysctl` 設定はデフォルトで 2 になり、`mmap` ベース、スタック、`VDSO` ページのランダム化に加えてヒープのランダム化も有効になります。

1996 年以前の古い `libc.so.5` バイナリとの互換性を保つため、このオプションはカーネルに組み込まれています。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

COMPAT_VDSO

この設定オプションは x86-64 に関連しますが、`aarch64` とは関係ありません。これを `n` に設定すると、Amazon Linux カーネルは 32 ビットの仮想化動的共有オブジェクト (VDSO) を予測可能なアドレスに表示しなくなります。このオプションを `n` に設定したことで障害が発生することがわかっている最新の `glibc` は 2004 年の `glibc 2.3.3` です。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

CONFIG_DEBUG ゲート強化

`CONFIG_DEBUG` ゲートの Linux カーネル設定オプションは通常、デバッグ用の問題を想定して構築されたカーネルで使用するよう設計されており、パフォーマンスなどは優先されません。AL2023 は `CONFIG_DEBUG_LIST` 強化オプションを有効にします。

IOMMU を設定する前に EFI スタブで PCI デバイスの DMA を無効にする

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

カーネルとユーザースペース間のメモリコピーの強化

カーネルがユーザースペースにメモリーをコピーしたり、ユーザー空間からメモリーをコピーしたりする必要がある場合、このオプションはいくつかのチェックを有効にすることで、ある種のヒープオーバーフローの問題を防ぐことができます。

`CONFIG_HARDENED_USERCOPY_FALLBACK` オプションはカーネル 4.16 から 5.15 までにあって、カーネルデベロッパーが不足している許可リストエントリを `WARN()` 経由で見つけやすくするためのものです。AL2023 には 6.1 カーネルが含まれているため、このオプションは AL2023 には関係ありません。

CONFIG_HARDENED_USERCOPY_PAGESPAN オプションは主にデベロッパー向けのデバッグオプションとしてカーネルに存在していましたが、AL2023 の 6.1 カーネルには適用されなくなりました。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

休止のサポート

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。[オンデマンドインスタンスを休止状態にしたり、中断されたスポットインスタンスを休止状態にしたりする](#)には、このオプションを有効にする必要があります。

乱数の生成

AL2023 カーネルは、EC2 内で適切なエントロピーが使用できるように設定されています。

CONFIG_INET_DIAG

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。AL2023 では、この機能はオプションで、カーネルモジュールとして構築されています。

割り当てと割り当て解除時にすべてのカーネルページとスラブアロケータメモリーをゼロにする

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。これらのオプションは、この機能をデフォルトで有効にするとパフォーマンスに影響する可能性があるため、AL2023 では無効になっています。CONFIG_INIT_ON_ALLOC_DEFAULT_ONinit_on_alloc=1 動作はカーネルコマンドラインに追加することで有効にでき、CONFIG_INIT_ON_FREE_DEFAULT_ON 動作は init_on_free=1 に追加することで有効にできます。

すべてのスタック変数をゼロ (CONFIG_INIT_STACK_ALL_ZERO) に初期化する

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。このオプションには GCC 12 以上が必要ですが、AL2023 には GCC 11 が含まれています。

カーネルモジュールの署名

AL2023 はカーネルモジュールの署名および署名の検証を行います。サードパーティモジュールを構築するユーザーの互換性を保つため、モジュールに有効な署名を要求する `CONFIG_MODULE_SIG_FORCE` オプションは有効化されていません。すべてのカーネルモジュールが署名されていることを確認したいユーザは、これを強制するように [Linux セキュリティモジュール \(LSM\) のロックダウン](#) を設定できます。

kexec

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。このオプションは `kdump` 機能を使用できるように有効になっています。

IOMMU サポート

AL2023 は IOMMU サポートを有効にします。`CONFIG_IOMMU_DEFAULT_DMA_STRICT` オプションはデフォルトでは有効になっていませんが、この機能はカーネルコマンドラインに `iommu.passthrough=0 iommu.strict=1` を追加することで設定できます。

kfence

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

従来の `pty` のサポート

AL2023 は最新の PTY インターフェイス (`devpts`) を使用します。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

Linux セキュリティモジュール (LSM) のロックダウン

AL2023 は `lockdown LSM` をビルドします。これにより、セキュアブートを使用するとカーネルが自動的にロックダウンされます。

`CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY` オプションは有効になっていません。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。セキュアブートを使用しない場合は、必要に応じてロックダウン LSM を有効に設定できます。

ページポイズニング

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。[割り当てと割り当て解除時にすべてのカーネルページとスラブアロケーターメモリをゼロにする](#)と同様に、AL2023 カーネルではパフォーマンスに影響する可能性があるため、このオプションは無効になっています。

スタックプロテクター

AL2023 カーネルは、`-fstack-protector-strong` オプションで有効になっている GCC のスタックプロテクター機能を搭載して構築されています。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

seccomp BPF API

seccomp 強化機能は、systemd およびコンテナランタイムなどのソフトウェアがユーザースペースアプリケーションを強化するために使用されます。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

panic() タイムアウト

AL2023 カーネルは、この値が 0 に設定された状態で設定されています。つまり、パニックが発生してもカーネルは再起動しません。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。これは `sysctl`、`/proc/sys/kernel/panic`、およびカーネルコマンドラインで設定できます。

セキュリティモデル

AL2023 ではデフォルトで許可モードの SELinux が有効になっています。詳細については、「[AL2023 の SELinux モードの設定](#)」を参照してください。

[Linux セキュリティモジュール \(LSM\) のロックダウン](#) および `yama` モジュールも有効になっています。

/proc/kcore

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

システムコールエントリでのカーネルスタックオフセットのランダム化

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。これはカーネルコマンドラインで `randomize_kstack_offset=on` を設定することで有効にできます。

参照カウントチェック (CONFIG_REFCOUNT_FULL)

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。このオプションはパフォーマンスに影響する可能性があるため、現時点では有効になっていません。

スケジューラーによる SMT コアの認識 (CONFIG_SCHED_CORE)

AL2023 カーネルは CONFIG_SCHED_CORE を使用して構築されているため、ユーザースペースのアプリケーションで `prctl(PR_SCHED_CORE)` を使用できます。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

`schedule()`(CONFIG_SCHED_STACK_END_CHECK) 呼び出しでスタックに問題がないか確認する

AL2023 カーネルは CONFIG_SCHED_STACK_END_CHECK が有効になっている状態で構築されています。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

メモリアロケータの強化

AL2023 カーネルで

は、CONFIG_SHUFFLE_PAGE_ALLOCATOR、CONFIG_SLAB_FREELIST_HARDENED、および CONFIG_SLAB_FREELIST_RANDOM オプションを使用してカーネルメモリアロケータを強化できます。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

SLUB デバッグのサポート

AL2023 カーネルは、このオプションによってカーネルコマンドラインで有効にできるアロケータのオプションのデバッグ機能を CONFIG_SLUB_DEBUG で有効にします。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

CONFIG_STATIC_USER_MODE_HELPER

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。これは、現在

Amazon Linux にはない配布からの特別なサポートがCONFIG_STATIC_USERMODEHELPER で必要のためです。

読み取り専用カーネルテキストおよび rodata (CONFIG_STRICT_KERNEL_RWX および CONFIG_STRICT_MODULE_RWX)

AL2023 カーネルは、カーネルとカーネルモジュールのテキストと rodata メモリを読み取り専用としてマークし、テキスト以外のメモリは実行不可としてマークするように設定されています。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の1つです。

TCP syncookie のサポート (CONFIG_SYN_COOKIES)

AL2023 カーネルは TCP syncookie をサポートするように構築されています。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の1つです。

ガードページ (CONFIG_VMAP_STACK) 付きの仮想化マップスタック

AL2023 カーネルは CONFIG_VMAP_STACK で構築されているため、ガードページ付きの仮想化マッピングカーネルスタックが可能になります。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の1つです。

コンパイラ警告をエラー (CONFIG_WERROR) として構築する

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の1つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

関数 exit (CONFIG_ZERO_CALL_USED_REGS) 時にレジスタをゼロにする

このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の1つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

ユーザースペース割り当ての最小アドレス

この強化オプションは、カーネル NULL ポインターのバグの影響を軽減するのに役立ちます。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の1つです。

clang 特定の強化オプション

AL2023 カーネルは clang ではなく GCC で構築されているため、CONFIG_CFI_CLANG 強化オプションを有効にすることはできません。CONFIG_CFI_PERMISSIVE も適用されません。このオプ

シヨンは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

x86-64 固有のカーネル強化オプション

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_AMD_IOMMU	該当なし	y	該当なし	y	該当なし	y
CONFIG_AMD_IOMMU_V2	該当なし	y	該当なし	該当なし	該当なし	該当なし
CONFIG_IA32_EMULATION	該当なし	y	該当なし	y	該当なし	y
CONFIG_INTEL_IOMMU	該当なし	y	該当なし	y	該当なし	y
CONFIG_INTEL_IOMMU_DEFAULT_ON	該当なし	n	該当なし	n	該当なし	n
CONFIG_INTEL_IOMMU_SVM	該当なし	n	該当なし	n	該当なし	n
CONFIG_LEGACY_VSYSCALL_NONE	該当なし	n	該当なし	n	該当なし	n

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_MODIFY_LDT_SYSCALL	該当なし	n	該当なし	n	該当なし	n
CONFIG_PAGE_TABLE_ISOLATION	該当なし	y	該当なし	該当なし	該当なし	該当なし
CONFIG_RANDOMIZE_MEMORY	該当なし	y	該当なし	y	該当なし	y
CONFIG_X86_64	該当なし	y	該当なし	y	該当なし	y
CONFIG_X86_64_MSR	該当なし	y	該当なし	y	該当なし	y
CONFIG_X86_64_VSYSCALL_EMULATION	該当なし	y	該当なし	y	該当なし	y
CONFIG_X86_64_X32	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_X86_64_X32_ABI	該当なし	n	該当なし	n	該当なし	n

x86-64 サポート

基本的な x86-64 サポートには、物理アドレス拡張 (PAE) と実行なし (NX) ビットのサポートが含まれます。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

AMD と Intel IOMMU のサポート

AL2023 カーネルは AMD と Intel IOMMU をサポートするように構築されています。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

CONFIG_INTEL_IOMMU_DEFAULT_ON オプションは設定されていませんが、カーネルコマンドラインに `intel_iommu=on` を渡すことで有効にできます。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

CONFIG_INTEL_IOMMU_SVM オプションは現在 AL2023 では有効化されていません。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

32 ビットユーザースペースのサポート

Important

32 ビット x86 ユーザースペースのサポートは廃止され、32 ビットユーザースペースバイナリの実行のサポートは Amazon Linux の今後のメジャーバージョンで削除される可能性があります。

Note

AL2023 には 32 ビットパッケージは含まれなくなりましたが、カーネルは 32 ビットユーザースペースの実行を引き続きサポートします。詳細については「[32 ビット x86 \(i686\) パッケージ](#)」を参照してください。

32 ビットユーザースペースアプリケーションの実行をサポートするため、AL2023 では CONFIG_X86_VSYSCALL_EMULATION オプションは有効にせず、CONFIG_IA32_EMULATION、CONFIG_COMPAT、CONFIG_X86_VSYSCALL_EMULATION オプションを有効にします。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

64 ビットプロセッサ用の x32 ネイティブ 32 ビット ABI は有効になっていません (CONFIG_X86_X32 および CONFIG_X86_X32_ABI)。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

x86 モデル固有レジスタ (MSR) のサポート

turbostat をサポートするため CONFIG_X86_MSR オプションは有効になっています。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

modify_ldt システムコール

AL2023 では、ユーザープログラムが modify_ldt システムコールを使用して x86 ローカル記述子テーブル (LDT) を変更することはできません。この呼び出しは 16 ビットコードまたはセグメント化されたコードを実行するために必要です。この呼び出しがないと、WINE でプログラムを実行したり、非常に古いスレッドライブラリを実行したりする場合など、dosemu などのソフトウェアが動作しなくなる可能性があります。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

ユーザーモードでカーネルマッピングを削除する

AL2023 は、カーネルアドレスの大部分がユーザースペースにマップされないようにカーネルを設定します。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

カーネルメモリセクションをランダム化する

AL2023 は、カーネルメモリセクションのベース仮想化アドレスをランダム化するようにカーネルを設定します。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

aarch64 固有のカーネル強化オプション

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_ARM64_BTI	y	該当なし	y	該当なし	y	該当なし

CONFIG オプション	AL2023/6.1/aarch64	AL2023/6.1/x86_64	AL2023/6.12/aarch64	AL2023/6.12/x86_64	AL2023/6.18/aarch64	AL2023/6.18/x86_64
CONFIG_ARM64_BTI_KERNEL	該当なし	該当なし	該当なし	該当なし	該当なし	該当なし
CONFIG_ARM64_PTR_AUTH	y	該当なし	y	該当なし	y	該当なし
CONFIG_ARM64_PTR_AUTH_KERNEL	y	該当なし	y	該当なし	y	該当なし
CONFIG_ARM64_SW_TTBR0_PAN	y	該当なし	y	該当なし	y	該当なし
CONFIG_UNMAP_KERNEL_AT_EL0	y	該当なし	y	該当なし	y	該当なし

ブランチターゲット識別

AL2023 カーネルでは、ブランチターゲット識別 (CONFIG_ARM64_BTI) がサポートされます。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

AL2023 で GCC を使用して構築された CONFIG_ARM64_BTI_KERNEL オプションは有効になっていません。また、[gcc のバグ](#)により、このオプションを使ったカーネル構築のサポートは、[アップストリームのカーネルでは現在無効になっています](#)。このオプションは[カーネルセルフプロテクションプロジェクト \(KSPP\) の推奨設定](#)の 1 つですが、AL2023 ではこの設定オプションを KSPP が推奨するものに設定していません。

ポインター認証 (CONFIG_ARM64_PTR_AUTH)

AL2023 カーネルは、リターン指向プログラミング (ROP) 手法の軽減に役立つポインター認証拡張機能 (ARMv8.3 拡張機能の一部) をサポートするように構築されています。[Graviton](#) でのポインター認証に必要なハードウェアサポートは Graviton 3 で導入されました。

CONFIG_ARM64_PTR_AUTH オプションは有効になっており、ユーザースペースのポインター認証をサポートします。CONFIG_ARM64_PTR_AUTH_KERNEL オプションも有効になっているため、AL2023 カーネルはリターンアドレス保護を自分で使うことができます。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

TTBR0_EL1 切り替えを決してしない特権アクセスをエミュレートする

このオプションは、ユーザーアクセスルーチンによって一時的にのみ有効な値に設定される TTBR0_EL1 で、カーネルがユーザー空間のメモリに直接アクセスすることを防ぎます。

このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

ユーザースペースでの実行時にカーネルをアンマップする

AL2023 カーネルは、ユーザースペース (CONFIG_UNMAP_KERNEL_AT_EL0) での実行時にカーネルをアンマップするように設定されています。このオプションは、[カーネルセルフプロテクションプロジェクト推奨設定](#)の 1 つです。

AL2023 でのリポジトリメタデータ署名

リリース以降 2023.11.20260406、AL2023 リポジトリにはリポジトリメタデータの暗号化署名が含まれています。各リポジトリの repomd.xml ファイルには、パッケージがダウンロードされる前にリポジトリメタデータの信頼性と整合性を検証するために使用できる、デタッチされた GPG 署名ファイル (repomd.xml.asc) が付属しています。

この署名は、個々のパッケージを検証する既存の RPM パッケージ署名 (gpgcheck) に追加されます。リポジトリメタデータ署名は、使用可能なパッケージのリストやチェックサムなど、リポジトリの内容を説明するメタデータを検証します。

リポジトリメタデータ署名の仕組み

AL2023 リポジトリが公開されると、リポジトリメタデータ (repomd.xml) は KMS AWS キーを使用して署名されます。デタッチされた署名 (repomd.xml.asc) は、リポジトリ内のメタデータと一緒に配置されます。

リポジトリ設定 `repo_gpgcheck` を有効にすると、 はリポジトリメタデータを使用する前に、GPG パブリックキーに対して `repomd.xml.asc` 署名DNFを自動的にダウンロードして検証します。署名の検証に失敗した場合、 はリポジトリメタデータDNFを拒否し、そのリポジトリからのパッケージオペレーションを続行しません。の詳細については `repo_gpgcheck`、[DNF「設定リファレンス」](#) を参照してください。

次の AL2023 リポジトリには、署名付きメタデータが含まれています。

- コアリポジトリ (`amazonlinux`)
- カーネル Livepatch リポジトリ (`kernel-livepatch`)
- NVIDIA リポジトリ (`amazonlinux-nvidia`)
- Amazon Linux リポジトリの補足パッケージ (`amazonlinux-spall`)

gpgcheck との違い repo_gpgcheck

設定	検証内容	AL2023 のデフォルト
<code>gpgcheck=1</code>	インストール前に個々の RPM パッケージの GPG 署名を検証します。	有効
<code>repo_gpgcheck=1</code>	リポジトリを使用する前に、リポジトリメタデータ (<code>repomd.xml</code>) の GPG 署名を検証します。	無効 (2023.12 四半期ごとのリリースからデフォルトで有効)

`gpgcheck` との両方を有効にすることを強くお勧めします `repo_gpgcheck`。これにより、リポジトリメタデータと個々のパッケージの両方が使用前に検証されます。

リポジトリメタデータ検証の有効化

設定ファイルを更新することで、個々のリポジトリのリポジトリメタデータ検証を有効にできます。

Important

2023.12 四半期ごとのリリース以降、 は AL2023 リポジトリ設定ファイルでデフォルトで有効 `repo_gpgcheck=1` になります。

特定のリポジトリに対して を有効にする

の AL2023 リポジトリ設定ファイルは`repo_gpgcheck=0`デフォルトで`/etc/yum.repos.d/`設定されています。リポジトリメタデータの検証を有効にするには、リポジトリ設定1でこの値を に変更します。たとえば、コアリポジトリに対して有効にするには:

```
[amazonlinux]
name=Amazon Linux 2023 repository
...
gpgcheck=1
repo_gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2023
```

リポジトリメタデータ署名が機能していることの確認

を有効にすると`repo_gpgcheck=1`、DNFキャッシュをクリアしてメタデータを更新することで、メタデータの検証が機能していることを確認できます。

```
[ec2-user ~]$ sudo dnf clean metadata
[ec2-user ~]$ sudo dnf makecache
```

メタデータの検証が成功すると、 は GPG キーDNFをインポートし (まだインポートされていない場合)、エラーなしでメタデータキャッシュを作成します。以下のような出力結果が表示されるはずです。

```
Amazon Linux 2023 repository          1.7 MB/s | 1.8 kB      00:00
Importing GPG key 0xD832C631:
  Userid      : "Amazon Linux <amazon-linux@amazon.com>"
  Fingerprint: B21C 50FA 44A9 9720 EAA7 2F7F E951 904A D832 C631
  From        : /etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2023
Amazon Linux 2023 repository          18 MB/s | 55 MB       00:03
Metadata cache created.
```

署名の検証に失敗した場合、 は GPG 署名の検証に失敗し、メタデータキャッシュの作成に失敗したことを示すエラーメッセージDNFを表示します。

AL2023 リポジトリの GPG パブリックキー

リポジトリメタデータの検証に使用される GPG パブリックキーは、対応するリポジトリ設定 RPMs にインストールされます `/etc/pki/rpm-gpg/`。次の表に、各リポジトリで使用されるパブリックキーを示します。

Repository	パッケージ署名キー	リポジトリデータ署名キー	で配布
コア (amazonlinux)	RPM-GPG-KEY-amazon -linux-2023	RPM-GPG-KEY-amazon -linux-2023	system-release
カーネル Livepatch (kernel-livepatch)	RPM-GPG-KEY-amazon -linux-2023	RPM-GPG-KEY-amazon -linux-2023	system-release
NVIDIA (amazonlinux-nvidia)	RPM-GPG-KEY-NVIDIA -D42D0685	RPM-GPG-KEY-amazon -linux-2023-nvidia	nvidia-release
SPAL (amazonlinux-spal)	RPM-GPG-KEY-amazon linux-spal	RPM-GPG-KEY-amazon linux-spal	spal-release

これらのキーは、対応するリポジトリ設定 RPM をインストールすると自動的にインストールされます。

AL2023 での UEFI セキュアブート

AL2023 は、リリース 2023.1 から UEFI セキュアブートをサポートしています。AL2023 は UEFI と UEFI セキュアブートの両方をサポートする Amazon EC2 インスタンスで使用する必要があります。詳細については、「Amazon EC2 ユーザーガイド」の「[UEFI ブートモードで Amazon EC2 インスタンスを起動するための要件](#)」を参照してください。

UEFI セキュアブートが有効になっている AL2023 インスタンスは、Amazon が署名したカーネルレベルのコード (Linux カーネルとモジュールを含む) のみを受け付けるため、インスタンスが AWS によって署名されたカーネルレベルのコードのみを実行するようにできます。

Amazon EC2 インスタンスと UEFI Secure Boot の詳細については、[「Amazon EC2 ユーザーガイド」の「Amazon EC2 インスタンスの UEFI Secure Boot」](#)を参照してください。Amazon EC2

前提条件

- AL2023 リリース 2023.1 以降の AMI を使用している必要があります。
- インスタンスタイプは UEFI Secure Boot をサポートする必要があります。詳細については、「Amazon EC2 ユーザーガイド」の[「UEFI ブートモードで Amazon EC2 インスタンスを起動するための要件」](#)を参照してください。

AL2023 で UEFI セキュアブートを有効にする

標準 AL2023 AMI には、ブートローダーおよびキーで署名されたカーネルが組み込まれています。UEFI セキュアブートを有効にするには、既存のインスタンスを登録するか、スナップショットからイメージを登録して UEFI セキュアブートがあらかじめ有効になっている AMI を作成します。デフォルトでは、UEFI セキュアブートは標準の AL2023 AMI で有効になっていません。

インスタンスタイプが UEFI をサポートしている場合、AL2023 AMI のブートモードは、これらの AMI で起動されたインスタンスが UEFI ファームウェアを使用するように uefi-preferred に設定されています。インスタンスタイプが UEFI をサポートしていない場合、インスタンスはレガシー BIOS ファームウェアで起動されます。インスタンスをレガシー BIOS モードで起動すると UEFI セキュアブートは実行されません。

Amazon EC2 インスタンスの AMI ブートモードの詳細については、「Amazon EC2 [ユーザーガイド](#)」の[Amazon EC2 ブートモードを使用したインスタンスの起動動作](#)」を参照してください。

Amazon EC2

トピック

- [既存のインスタンスの登録](#)
- [スナップショットからイメージを登録する](#)
- [失効の更新](#)
- [AL2023 での UEFI 安全ブートのしくみ](#)
- [独自のキーを登録する](#)

既存のインスタンスの登録

既存のインスタンスを登録するには、特定の UEFI ファームウェア変数に、ファームウェアがブートローダーを検証し、ブートローダーが次の起動時にカーネルを検証できるようにする一連のキーを設定します。

1. Amazon Linux には、登録プロセスを簡素化するツールが用意されています。以下のコマンドを実行して、必要なキーと証明書のセットをインスタンスにプロビジョニングします。

```
sudo amazon-linux-sb enroll
```

2. 以下のコマンドを実行して、インスタンスを再起動します。インスタンスが再起動すると、UEFI セキュアブートが有効になります。

```
sudo reboot
```

Note

Amazon Linux AMI は現在 Nitro Trusted Platform Module (NitroTPM) をサポートしていません。UEFI セキュアブートに加えて NitroTPM が必要な場合は、以下のセクションの情報を参照します。

スナップショットからイメージを登録する

Amazon EC2 register-image API を使用して Amazon EBS ルートボリュームのスナップショットから AMI を登録する場合、UEFI 変数ストアの状態を含むバイナリブロッブを使用して AMI をプロビジョニングできます。AL2023 UefiData を提供することで UEFI セキュアブートが有効になり、前のセクションの手順を実行する必要がなくなります。

バイナリ BLOB の作成と使用の詳細については、「Amazon EC2 ユーザーガイド」の「[あらかじめ入力された変数ストアを含むバイナリ BLOB を作成する](#)」を参照してください。

AL2023 には、Amazon EC2 インスタンスで直接使用できる構築済みのバイナリ BLOB が用意されています。バイナリ BLOB は実行中のインスタンスの `/usr/share/amazon-linux-sb-keys/uefi.vars` にあります。この BLOB は、リリース 2023.1 以降の AL2023 AMI にデフォルトでインストールされている `amazon-linux-sb-keys RPM` パッケージによって提供されます。

Note

キーおよび失効のバージョンが最新であることを確認するには、AMI の作成に使用したのと同じリリースの AL2023 の BLOB を使用します。

イメージの登録時には、[RegisterImage](#) API の `BootMode` パラメータを `uefi` に設定することをお勧めします。これにより、`TpmSupport` パラメータを `v2.0` に設定することで NitroTPM を有効にできます。また、UEFI をサポートしないインスタンスタイプに切り替えたときに UEFI セキュアブートが有効になり、誤って無効にされないように `BootMode` を `uefi` に設定できます。

NitroTPM の詳細については、[Amazon EC2 ユーザーガイド](#) の「[Amazon EC2 インスタンスの NitroTPM](#)」を参照してください。Amazon EC2

失効の更新

Amazon Linux は、更新されたキーで署名された新しいバージョンのブートローダー `grub2` または Linux カーネルを配布する必要がある場合があります。その場合は、以前のバージョンのブートローダーの悪用可能なバグが UEFI セキュアブート検証プロセスをバイパスしてしまうことを防ぐために、古いキーを取り消す必要がある場合があります。

`grub2` または `kernel` パッケージにパッケージを更新すると、実行中のインスタンスの UEFI 変数ストアへの失効リストが常に自動的に更新されます。つまり、UEFI セキュアブートを有効にすると、パッケージのセキュリティ更新をインストールした後は、古いバージョンのパッケージを実行できなくなります。

AL2023 での UEFI 安全ブートのしくみ

他の Linux 配布とは異なり、Amazon Linux には第 1 ステージのブートローダーとして機能するシムと呼ばれる追加コンポーネントはありません。シムは通常、Microsoft のキーで署名されています。例えば、シム付きの Linux ディストリビューションでは、シムが `grub2` ブートローダーをロードし、シム独自のコードを使用して Linux カーネルを検証します。さらに、シムは UEFI 変数ストアにある Machine Owner Key (MOK) データベースに独自のキーセットと失効情報を保持し、`mokutil` ツールで制御します。

Amazon Linux にはシムはありません。AMI の所有者が UEFI 変数を制御するため、この中間ステップは不要で、起動時間と起動時間に悪影響を及ぼします。また、望ましくないバイナリが実行される可能性を減らすため、デフォルトではベンダーキーへの信頼を含めないようにしました。通常通り、必要に応じてバイナリを含めることができます。

Amazon Linux では、UEFI が grub2 ブートローダーを直接ロードして検証します。grub2 ブートローダーは、ロード後に UEFI を使用して Linux カーネルを検証するように変更されました。そのため、Linux カーネルは、通常の UEFI db 変数 (認証キーデータベース) に格納されているのと同じ証明書を使用して検証され、ブートローダーや他の UEFI バイナリと同じ dbx 変数 (失効データベース) に対してテストされます。db データベースおよび dbx データベースへのアクセスを制御する独自の PK キーおよび KEK キーを提供しているため、シムなどの仲介なしで、必要に応じて署名付きの更新と失効を配布できます。

UEFI セキュアブートの詳細については、[Amazon EC2 ユーザーガイド](#)の「[UEFI セキュアブートが Amazon EC2 インスタンスと連携する方法](#)」を参照してください。Amazon EC2

独自のキーを登録する

前のセクションで説明したように、Amazon Linux では Amazon EC2 で shim が UEFI セキュアブートを実行する必要はありません。他の Linux 配布ドキュメントを読んでいると、AL2023 にはない、mokutil を使用した Machine Owner Key (MOK) データベースの管理に関するドキュメントを見かける可能性があります。shim および MOK 環境は、Amazon EC2 が UEFI セキュアブートを実装する方法には当てはまらない UEFI ファームウェアへのキー登録のいくつかの制限を回避します。Amazon EC2 には、UEFI 変数ストア内のキーを簡単に直接操作するメカニズムがあります。

独自のキーを登録したい場合は、既存のインスタンス内の変数ストアを操作するか (「[インスタンス内から変数ストアにキーを追加する](#)」を参照してください)、事前入力されたバイナリ BLOB を構築できます (「[事前入力された変数ストアを含むバイナリ BLOB の作成](#)」を参照してください)。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。