



사용 설명서

Amazon Linux 2023



Amazon Linux 2023: 사용 설명서

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

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 더불어 Amazon 브랜드 이미지를 실추시키거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

Amazon Linux 2023이란 무엇입니까?	1
릴리스 케이던스	1
메이저 및 마이너 릴리스	2
새 릴리스 사용	2
장기 지원 정책	2
이름 지정 및 버전 관리	3
성능 및 운영 최적화	4
Fedora와의 관계	5
사용자 지정 cloud-init	5
보안 업데이트 및 기능	7
관리 업데이트	7
클라우드 보안	7
SELinux 모드	7
규정 준수 프로그램	8
SSH 서버 기본값	8
OpenSSL 3의 주요 기능	8
네트워킹 서비스	8
코어 툴체인 패키지 glibc, gcc 및 binutils	9
패키지 관리 도구	10
SSH 서버 기본 구성	10
더 이상 사용되지 않는 기능	12
compat- 패키지	12
AL1에서 더 이상 사용되지 않는 기능이 중단되고 AL2에서 제거됨	12
32비트 x86(i686) AMI	13
aws-apitools-*가 AWS CLI로 대체	13
systemd가 AL2에서 upstart를 대체	14
AL2에서는 기능이 더 이상 사용되지 않고 AL2023에서는 제거됨	14
32비트 x86(i686) 패키지	15
aws-apitools-* 로 대체됨 AWS CLI	15
amazon-cloudwatch-agent를 대체합니다. awslogs	16
bzip 개정 제어 시스템	16
cgroup v1	16
log4j 핫패치(log4j-cve-2021-44228-hotpatch)	16
lsb_release 및 system-lsb-core 패키지	17

mcrypt	17
OpenJDK 7(java-1.7.0-openjdk)	18
Python 2.7	18
rsyslog-openssl를 대체합니다. rsyslog-gnutls	18
네트워크 정보 서비스(NIS) / yp	18
Amazon VPC create-dhcp-options의 여러 도메인 이름	18
Sun RPC의 glibc	19
audit 로그의 OpenSSH 키 지문	19
ld.gold 링커	19
ping6	20
ftp 패키지	20
AL2023에서 사용 중단	21
32비트 x86(i686) 런타임 지원	22
aspell	22
Berkeley DB(libdb)	22
cron	23
IMDSv1	23
pcre 버전 1	23
System V init (sysvinit)	24
EOL 패키지는 더 이상 사용되지 않습니다.	24
AL2와 AL2023 비교	25
패키지 추가, 업그레이드 및 삭제	26
각 릴리스에 대한 지원	26
이름 지정 및 버전 변경	26
최적화	26
다수 업스트림 제공	27
네트워킹 시스템 서비스	27
패키지 관리자	27
cloud-init 사용하기	27
데스크톱 그래픽 지원	28
컴파일러 트리플렛	28
32비트 x86 (i686) 패키지	28
lsb_release 및 system-lsb-core 패키지	28
EPEL	29
axel - HTTP/FTP 클라이언트	31
brotli 및 libbrotli - 압축	31

collectd - 통계 수집 대몬	32
cpulimit	32
exim - 우편 전송 에이전트	32
fuse3 - 사용자 공간의 파일 시스템(FUSE) v3	32
ganglia - 분산 모니터링 시스템	32
git-lfs - Git을 사용하여 대용량 파일 버전 관리	33
haveged - HAVEGE 알고리즘을 사용하는 엔트로피 소스	33
inotify-tools - inotify 명령줄 도구	33
iperf - TCP/UDP 성능 벤치마크	33
jemalloc - 대체 malloc 구현	34
libbsd - BSD 호환 함수 라이브러리	34
libserf - HTTPS 클라이언트 라이브러리	34
libzstd - zstd 압축 라이브러리	34
lighttpd 웹 서버	35
lshell - 제한된 셸	35
monit - 프로세스, 파일, 디렉터리 및 디바이스 모니터	35
nodejs	35
perl-Config-General	36
python2-lockfile - 파일 잠금	36
python2-rsa - 순수 Python RSA	37
python2-simplejson - Python 2에 대한 JSON 루틴	37
rkhunter - Rootkit Hunter	37
rssh - OpenSSH와 함께 사용할 수 있는 제한된 셸	38
sscg - 자체 서명된 SSL 인증서 생성기	38
stress - 스트레스 테스트	38
stress-ng - 스트레스 테스트	38
tmpwatch - 마지막으로 액세스한 시간을 기준으로 파일 제거	38
xmlstarlet - 명령줄 XML 유틸리티	38
Python 2.7은 Python 3으로 대체되었습니다.	39
보안 업데이트	39
SELinux	39
OpenSSL 3	40
IMDSv2	40
log4j 핫패치 제거 (log4j-cve-2021-44228-hotpatch)	41
안정성을 위한 결정적 업그레이드	41
기본 Amazon EBS 볼륨 유형인 gp3	41

통합 제어 그룹 계층 구조 (cgroup v2)	42
systemd 타이머 교체 cron	42
향상된 도구 체인: gcc, binutils 및 glibc	42
systemd 저널 대체 rsyslog	43
패키지 종속성 최소화	44
curl 및 libcurl 패키지 변경	44
GNU 프라이버시 가드(GNUPG)	44
기본 JVM으로 Amazon Corretto 설정	45
AWS CLIv2	45
UEFI 기본 설정 및 보안 부팅	45
SSH 서버 기본 구성 변경	45
AL2 이후 AL2023 커널 변경 사항	46
IPv4 TTL	46
보안 중심으로 커널 구성 변경	46
기타 커널 구성 변경	51
커널 파일 시스템 지원	52
/tmp 변경	58
AMI 및 컨테이너 이미지 변경	58
Amazon Linux 2 및 AL2023 AMI 비교	58
Amazon Linux 2 및 AL2023 미니멀 AMI 비교	91
Amazon Linux 2 및 AL2023 컨테이너 비교	112
AL1과 AL2023 비교	120
각 릴리스에 대한 지원	120
systemd는 upstart를 init 시스템으로 대체합니다.	120
Python 2.6 및 2.7은 Python 3으로 대체되었습니다.	121
OpenJDK 8은 가장 오래된 JDK	121
AL1 이후 AL2023 커널 변경 사항	121
커널 라이브 패치	121
커널 파일 시스템 지원	121
보안에 중점을 둔 커널 구성 변경	124
기타 커널 구성 변경	127
AL1과 AL2023 AMI 비교	128
AL1과 AL2023 미니멀 AMI 비교	162
AL1과 AL2023 컨테이너 비교	182
시스템 사양	191
AL2023 실행을 위한 CPU 요구 사항	191

AL2023 ARM CPU 사양	191
AL2023 x86-64 CPU 사양	192
AL2023 실행을 위한 메모리(RAM) 요구 사항	192
Amazon Linux 커널	194
커널 수명 주기	194
커널 업데이트	195
배포 간 커널 버전 중첩	195
CVE 처리	196
해야 할 일	196
그래픽 데스크톱	197
관련 주제	197
Amazon Linux용 보조 패키지	198
Amazon Linux(또는 SPAL)용 보조 패키지란 무엇입니까?	198
이점	198
SPAL 패키지 지원	199
패키지 관련 문제 보고	199
관련 주제	200
AL2023에서 SPAL 구성	200
사전 조건	200
사전 조건 확인	200
시스템에 SPAL 설치	201
SPAL 패키지 설치	202
시스템에서 SPAL 리포지토리 제거	203
관련 주제	203
애플리케이션 실행	205
systemd를 사용한 리소스 제어	205
일회성 명령 실행을 위한 systemd-run을 사용한 리소스 제어	205
systemd 서비스의 리소스 제어	208
cgroups 유틸리티 사용	212
에서 AL2023 사용 AWS	215
시작하기 AWS	215
에 가입 AWS 계정	215
관리자 액세스 권한이 있는 사용자 생성	216
프로그래밍 방식 액세스 권한 부여	217
Amazon EC2 AL2023	219
Amazon EC2 콘솔에서 AL2023 시작	220

SSM 파라미터 및를 사용하여 AL2023 시작 AWS CLI	221
를 사용하여 최신 AL2023 AMI 시작 CloudFormation	222
특정 AMI ID로 AL2023 실행	223
AL2023 AMI 사용 중단 및 수명 주기	224
AL2023 인스턴스에 연결	224
AL2023 표준(기본) AMI와 미니멀 AMI 비교	225
AL2023 컨테이너	253
AL2023 기본 컨테이너 이미지	253
AL2023 최소 컨테이너 이미지	255
기본 AL2023 컨테이너 이미지 빌드	257
AL2023 컨테이너 이미지 패키지 목록 비교	261
AL2023 미니멀 AMI와 컨테이너 이미지 비교	267
AL2023 Elastic Beanstalk	284
AL2023 CloudShell	284
AL2023 Amazon ECS 컨테이너 호스트	285
AL2 이후 Amazon ECS 관련 변경 사항	285
Amazon ECS 최적화 AMI 사용자 지정	286
AL2023 기반 Amazon EFS	286
amazon-efs-utils	287
Amazon EFS 파일 시스템 마운트	287
AL2023 Amazon EMR	287
AL2023 기반 Amazon EMR 릴리스	287
EKS 기반 Amazon EMR에 AL2023 지원	288
의 AL2023 AWS Lambda	288
provided.al2023 Lambda 런타임	288
AL2023 기반 런타임	288
자습서	289
AL2023에 LAMP 설치	289
1단계: LAMP 서버 준비	290
2단계: LAMP 서버 테스트	294
3단계: 데이터베이스 서버 보안 설정	296
4단계: (선택 사항) phpMyAdmin 설치	297
문제 해결	300
관련 주제	301
AL2023에서 SSL/TLS 구성	301
사전 조건	302

1단계: 서버에서 TLS 활성화	303
2단계: CA가 서명한 인증서 가져오기	306
3단계: 보안 구성 테스트 및 하드닝	313
문제 해결	317
AL2023에서 WordPress 블로그 호스팅	317
사전 조건	318
WordPress 설치	319
다음 단계	329
도움말! 내 퍼블릭 DNS 이름이 변경되어 블로그를 사용할 수 없습니다.	330
Redis 6에서 AL2023의 Valkey로 전환	331
Redis 6 지원 타임라인	331
Valkey 소개	331
마이그레이션 계획 및 타임라인	331
마이그레이션 옵션 및 단계	332
관련 주제	335
AL2023에 GNOME 설치	335
사전 조건	335
설치	336
관련 주제	336
AL2023에서 VNC 구성	336
사전 조건	337
1단계: 설치	337
2단계: 구성	337
3단계: VNC 클라이언트를 사용하여 연결	339
(선택 사항) 부팅 시 서비스 시작	339
(선택 사항) 유휴 잠금 화면 비활성화	340
관련 주제	340
AL2023 커널에서 Multi-Gen LRU(MGLRU) 사용	340
구성 및 튜닝	341
AL2023에 LibreOffice 설치	342
사전 조건	343
1단계: 시스템 아키텍처 결정	343
2단계: LibreOffice RPM 패키지 다운로드	343
3단계: 아카이브 추출	344
4단계: RPM 패키지 설치	344
5단계: 설치 확인	345

LibreOffice 시작	345
문제 해결	346
Amazon EC2 외부 AL2023	347
AL2023 VM 이미지 다운로드	347
지원되는 구성	347
KVM 조건	348
VMware 요구 사항	350
Hyper-V 요구 사항	352
AL2023 VM 구성	354
NoCloud seed.iso 기반 구성	355
VMware guestinfo 기반 구성	358
표준 AMI 및 KVM 이미지에 대한 AL2023 패키지 목록 비교	360
표준 AMI 및 VMware OVA 이미지에 대한 AL2023 패키지 목록 비교	385
표준 AMI 및 Hyper-V 이미지에 대한 AL2023 패키지 목록 비교	410
Amazon Linux 버전 식별	436
/etc/os-release	436
주요 차이점	436
필드 유형	437
/etc/os-release 예제	438
다른 배포판과 비교	440
Amazon Linux 특징	442
/etc/system-release	442
/etc/image-id	443
Amazon Linux 특징 예제	443
예제 코드	445
파일 시스템 레이아웃	459
/	459
/boot	460
/boot/efi	460
/etc	460
/home	460
/root	461
/srv	461
/tmp	461
/run	463
/usr	463

/usr/bin	463
/usr/include	463
/usr/lib'and'/usr/lib64	464
/usr/local	464
/usr/share	464
/var	464
/var/cache	464
/var/lib	464
/var/log	465
/var/spool	465
/var/tmp	465
AL2023 업데이트	467
업데이트를 안전하게 배포하기 위한 모범 사례	467
마이너 업데이트 준비	470
메이저 업데이트 준비	470
새 업데이트 알림 받기	471
버전 관리 리포지토리를 통한 결정적 업그레이드	472
메이저 및 마이너 릴리스에 포함된 업데이트를 관리하세요.	472
메이저 버전과 마이너 버전 업그레이드의 차이점은 다음과 같습니다.	472
업데이트를 사용할 수 있는 시기 알기	473
AL2023 리포지토리에서 사용할 수 있는 패키지 업데이트 관리	473
인스턴스 교체	474
현재 위치 결정적 업그레이드	474
업데이트 관리	482
사용 가능한 패키지 업데이트 확인	482
DNF 및 리포지토리 버전에서 보안 업데이트	487
(보안) 업데이트 후 자동 서비스 재시작	499
보안 업데이트를 적용하려면 언제 재부팅해야 하나요?	500
최신 리포지토리 버전이 활성화된 상태로 인스턴스 시작	501
패키지 지원 정보 가져오기	501
dnf check-release-update	502
새 리포지토리 추가, 사용 또는 사용 중지	505
cloud-init를 사용하여 리포지토리 추가	508
커널 라이브 패치	509
제한 사항	510
지원되는 구성 및 필수 조건	510

커널 라이브 패치 작업	511
커널 업데이트	516
AL2023의 Linux 커널 버전	516
AL2023을 최신 커널 버전으로 업데이트	517
AL2023 커널 - 자주 묻는 질문	521
프로그래밍 언어 및 런타임	523
C/C++ 및 포트란	523
GCC 14	524
언어 표준 버전 비교	525
.NET	527
.NET 최신 버전으로 마이그레이션	527
.NET 설치하기	527
Go	527
AL2023 Lambda 함수: Go	528
Java	528
Node.js	35
Perl	530
Perl 모듈	530
PHP	530
PHP 최신 버전으로 마이그레이션	530
PHP 7.x에서 마이그레이션	531
PHP 모듈	531
Python	531
Python 최신 버전으로 마이그레이션	532
Python 모듈	532
Ruby	532
Rust	534
AL2023 Lambda 함수: Rust	535
Swift	535
AL2023 함수: Swift	535
TypeScript	536
AL2023 예약 사용자 및 그룹	540
AL2023 예약 사용자 목록	540
AL2023 예약 그룹 목록	548
AL2023에서 사용 가능한 코덱	561
보안 및 규정 준수	563

보안 권고	564
ALAS 공지	564
ALAS FAQ	564
ALAS 권고	564
권고 사항 및 RPM 리포지토리	565
권고 ID	565
권고 릴리스 날짜 및 권고 업데이트 날짜	566
권고 유형	566
권고 심각도	566
권고 및 패키지	567
권고 및 CVE	568
권고 텍스트	568
커널 라이브 패치 권고 사항	569
updateinfo.xml 스키마	570
적용 가능한 권고 사항 나열	570
현재 위치 업데이트	574
권고에 언급된 업데이트 적용	574
AL2023의 SELinux 모드 설정	578
AL2023의 기본 SELinux 상태 및 모드	578
enforcing 모드로 변경	579
SELinux 비활성화 옵션	580
AL2023에서 FIPS 모드 활성화	582
AL2023 컨테이너에서 FIPS 모드 활성화	583
AL2023에서 OpenSSL FIPS 공급자 전환	585
커널 하드닝	587
커널 하드닝 옵션 (독립 아키텍처)	587
x86-64 전용 커널 하드닝 옵션	603
aarch64 전용 커널 하드닝 옵션	607
리포지토리 메타데이터 서명	608
리포지토리 메타데이터 서명 작동 방식	608
gpgcheck와의 차이점 repo_gpgcheck	609
리포지토리 메타데이터 확인 활성화	609
리포지토리 메타데이터 서명이 작동하는지 확인	610
AL2023 리포지토리를 GPG 퍼블릭 키	610
AL2023의 UEFI 보안 부팅	611
AL2023에서 UEFI 보안 부팅 활성화	612

기존 인스턴스 등록	612
스냅샷 이미지 등록	613
폐기 업데이트	613
AL2023에서 UEFI 보안 부팅 작동 방식	614
자체 키 등록	614
.....	dcxv

Amazon Linux 2023이란 무엇입니까?

Amazon Linux 2023(AL2023)은 Amazon Web Services(AWS)의 차세대 Amazon Linux입니다. AL2023을 사용하면 안전하고 안정적인 고성능 런타임 환경에서 클라우드 및 엔터프라이즈 애플리케이션을 개발하고 실행할 수 있습니다. 또한 Linux의 최신 혁신 기술에 대한 액세스와 함께 장기 지원을 제공하는 애플리케이션 환경을 사용할 수 있습니다. AL2023은 무료입니다.

AL2023은 Amazon Linux 2(AL2)의 후속 버전입니다. AL2023과 AL2의 차이점에 대한 자세한 내용은 [AL2와 AL2023 비교](#) 및 [AL2023 패키지 변경 사항](#) 섹션을 참조하세요.

주제

- [릴리스 케이던스](#)
- [이름 지정 및 버전 관리](#)
- [성능 및 운영 최적화](#)
- [Fedora와의 관계](#)
- [사용자 지정 cloud-init](#)
- [보안 업데이트 및 기능](#)
- [네트워킹 서비스](#)
- [코어 툴체인 패키지 glibc, gcc 및 binutils](#)
- [패키지 관리 도구](#)
- [SSH 서버 기본 구성](#)

릴리스 케이던스

Amazon Linux 2023(AL2023)은 2023년 3월에 릴리스되었으며 2029년 6월 30일까지 지원됩니다. 지원에는 두 단계가 있습니다.

- 표준 지원 – 이 단계에서 릴리스는 분기별 마이너 버전 업데이트를 받습니다. 표준 지원 단계는 2027년 6월 30일에 종료됩니다.
- 유지 관리 – 이 단계에서 릴리스는 보안 업데이트와 중요한 버그 수정만 받습니다. 이러한 업데이트는 사용 가능한 즉시 게시됩니다. 유지 관리 단계는 2029년 6월 30일에 종료됩니다.

메이저 및 마이너 릴리스

Amazon Linux(메이저 버전, 마이너 버전 또는 보안 릴리스)를 릴리스 할 때마다 새로운 Linux Amazon Machine Image(AMI)를 릴리스합니다.

- 메이저 버전 출시 — 스택 전반의 보안 및 성능에 대한 새로운 기능과 향상된 기능이 포함되어 있습니다. 개선 사항에는 커널, 툴체인 Glib C. OpenSSL. 기타 시스템 라이브러리 및 유틸리티의 주요 변경 사항이 포함될 수 있습니다. Amazon Linux 메이저 릴리스는 부분적으로 업스트림 Fedora Linux 배포판을 기반으로 합니다. AWS는 Fedora 아닌 특정 업스트림 패키지를 추가하거나 교체할 수 있습니다.
- 마이너 버전 릴리스 - 보안 업데이트, 버그 수정, 새로운 기능 및 패키지가 포함된 분기별 업데이트입니다. 각 마이너 버전은 새로운 기능 및 패키지 외에도 보안 및 버그 수정이 누적된 업데이트 목록입니다. 이러한 릴리스에는 PHP 같은 최신 언어 런타임이 포함될 수 있습니다. Ansible 및 Docker와 같은 유명 소프트웨어 패키지도 포함될 수 있습니다.

새 릴리스 사용

업데이트는 새로운 Amazon Machine Image(AMI) 릴리스 및 이에 해당하는 새로운 리포지토리 조합으로 제공됩니다. 기본적으로 새 AMI와 해당 리포지토리가 한 쌍이 됩니다. 하지만 시간이 지나면 최신 리포지토리 버전에 Amazon EC2 인스턴스를 실행하여 업데이트할 수 있습니다. 최신 AMI에 새 인스턴스를 실행하여 업데이트할 수도 있습니다.

장기 지원 정책

Amazon Linux는 모든 패키지에 대한 업데이트를 제공하고 Amazon Linux를 기반으로 구축된 애플리케이션 메이저 버전은 호환성을 유지합니다. glibc 라이브러리, OpenSSL, OpenSSH 및 DNF 패키지 관리자와 같은 핵심 패키지는 주요 AL2023 릴리스를 사용하는 동안 지원합니다. 코어 패키지에 포함되지 않은 패키지는 특정 업스트림 소스 기반에서 지원합니다. 다음 명령을 실행하여 개별 패키지의 구체적인 지원 상태 및 날짜를 확인할 수 있습니다.

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

다음 명령을 실행하여 현재 설치한 패키지 정보를 알 수 있습니다.

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


핵심 패키지 전체 목록은 미리 보기 중에 확정됩니다. 핵심 패키지에 포함된 패키지를 더 보고 싶다면 문의해 주세요. 피드백을 수집하여 평가합니다. AL2023 관련 피드백은 지정 AWS 담당자가 제공하거나 또는 GitHub의 [amazon-linux-2023 리포지토리](#)에 제출된 이슈로 받아볼 수 있습니다.

이름 지정 및 버전 관리

AL2023은 기본 2년 지원 동안 3개월마다 마이너 릴리스를 제공합니다. 각 릴리스는 0에서 N까지 증가하여 식별됩니다. 0은 해당 반복의 주요 릴리스를 나타냅니다. 모든 릴리스의 이름은 Amazon Linux 2023입니다. Amazon Linux 후속 버전이 릴리스되면 AL2023은 연장 지원 단계로 전환되며 보안 업데이트 및 중요한 버그 수정에 대한 업데이트를 받게 됩니다.

예를 들어, AL2023 마이너 릴리스의 형식은 다음과 같습니다.

- 2023.0.20230301
- 2023.1.20230601
- 2023.2.20230901

해당 AL2023 AMI는 다음 형식을 취합니다.

- 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

특정 마이너 버전의 정규 AMI 릴리스는 AMI 릴리스 날짜에 발생합니다.

- 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 또는 AL2023 인스턴스를 식별하는 권장 방법은 /etc/system-release-cpe에서 공통 플랫폼 목록(CPE) 문자열을 읽는 것을 권장합니다. 그런 다음 문자열을 해당 필드로 분할합니다. 마지막으로 플랫폼 및 버전 값을 확인합니다.

또한 AL2023에 플랫폼 식별을 위한 새 파일이 도입되었습니다.

- /etc/system-release에 대한 심볼릭 링크 /etc/amazon-linux-release

- `/etc/system-release-cpe`에 대한 심볼릭 링크 `/etc/amazon-linux-release-cpe`

이 두 파일은 인스턴스가 Amazon Linux임을 나타냅니다. 특정 플랫폼 및 버전 값을 알고 싶지 않다면 파일을 읽거나 문자열을 필드로 분할하지 않아도 됩니다.

성능 및 운영 최적화

Amazon Linux 6.1 커널

- AL2023은 Elastic Network Adapter(ENA) 및 Elastic Fabric Adapter(EFA) 디바이스의 최신 드라이버를 사용합니다. AL2023은 Amazon EC2 인프라의 하드웨어를 위한 성능 및 기능 백포트에 중점을 둡니다.
- x86_64 및 aarch64 인스턴스 유형에 커널 라이브 패치를 사용할 수 있습니다. 이렇게 하면 자주 재부팅하지 않아도 됩니다.
- 모든 커널 빌드 및 런타임 구성에는 AL2와 동일한 성능 및 운영 최적화가 다수 포함되어 있습니다.

기본 튜체인 선택 및 기본 빌드 플래그

- AL2023 패키지는 컴파일러 최적화(`-O2`)가 기본적으로 활성화된 상태로 빌드됩니다.
- AL2023 패키지를 빌드하려면 x86-64 시스템(`-march=x86-64-v2`)에 x86-64v2, aarch64 (`-march=armv8.2-a+crypto -mtune=neoverse-n1`)에 Graviton 2 이상 버전이 필요합니다.
- AL2023 패키지는 자동 벡터화가 활성화된 상태로 빌드됩니다(`-ftree-vectorize`).
- AL2023 패키지는 링크 시간 최적화(LTO)가 활성화된 상태로 빌드됩니다.
- AL2023은 업데이트된 Rust, Clang/LLVM, 및 Go 버전을 사용합니다.

패키지 선택 및 버전

- 주요 시스템 구성 요소에 대한 일부 백포트에는 Amazon EC2 인프라, 특히 Graviton 인스턴스에서 실행하기 위한 몇 가지 성능 개선 사항이 포함되어 있습니다.
- AL2023에 여러 AWS 서비스와 기능이 통합되어 있습니다. 여기에는 AWS CLI, SSM 에이전트, Amazon Kinesis 에이전트 및 CloudFormation이 포함되어 있습니다.
- AL2023에서 Amazon Corretto를 Java 개발 키트(JDK)로 사용할 수 있습니다.
- AL2023은 업스트림 프로젝트에서 최신 버전이 출시될 때 데이터베이스 엔진 및 프로그래밍 언어 런타임 업데이트를 제공합니다. 출시되면 최신 버전이 포함된 프로그래밍 언어 런타임이 추가됩니다.

클라우드 환경 배포

- 기본 AL2023 AMI 및 컨테이너 이미지는 패치 인스턴스 교체를 위해 자주 업데이트됩니다.
- 커널 업데이트는 AL2023 AMI 업데이트에 포함되어 있습니다. 즉, 커널을 업데이트하기 위해 `yum update` 및 `reboot` 같은 명령을 사용하지 않아도 됩니다.
- 표준 AL2023 AMI 외에도 미니멀 AMI 및 컨테이너 이미지도 사용할 수 있습니다. 서비스를 실행하는데 필요한 최소한의 패키지로 환경을 실행하려면 미니멀 AMI를 선택하세요.
- 기본적으로 AL2023 AMI 및 컨테이너는 패키지 리포지토리의 특정 버전에서만 실행됩니다. 실행 시 자동 업데이트가 되지 않습니다. 즉, 패키지 업데이트 시기를 관리할 수 있습니다. 프로덕션으로 출시하기 전에 언제든지 베타/감마 환경에서 테스트할 수 있습니다. 문제가 있는 경우 사전 검증된 롤백 경로를 이용할 수 있습니다.

Fedora와의 관계

AL2023은 Fedora와 독립적으로 자체 릴리스 및 지원 수명 주기를 유지합니다. AL2023은 오픈 소스 소프트웨어 업데이트, 더욱 다양한 패키지, 신속한 릴리스를 제공합니다. 이렇게 하면 익숙한 RPM 기반 운영 체제를 유지할 수 있습니다.

정식 출시된 AL2023 (GA) 버전은 특정 Fedora 릴리스와 직접 비교할 수 없습니다. AL2023 GA 버전에는 Fedora 34, 35, 36의 구성 요소가 포함되어 있습니다. 일부 구성 요소는 Fedora 구성 요소와 동일하며 일부는 수정되었습니다. 다른 구성 요소는 CentOS Stream 9의 구성 요소와 더 유사하거나 독립적으로 개발되었습니다. Amazon Linux 커널은 Fedora와는 별개로 선택된 kernel.org에 있는 장기 지원 옵션에서 가져왔습니다.

사용자 지정 cloud-init

cloud-init 패키지는 클라우드 컴퓨팅 환경에서 Linux 이미지 부트스트랩을 수행하는 오픈 소스 애플리케이션입니다. 자세한 내용은 [cloud-init 설명서](#)를 참조하세요.

AL2023에 cloud-init 사용자 지정 버전이 포함되어 있습니다. cloud-init를 사용하면 부팅 시 인스턴스 작업을 지정할 수 있습니다.

인스턴스를 실행 시 사용자-데이터 필드를 통해 cloud-init에 작업이 전달됩니다. 즉, 여러 사용 사례에 공통 Amazon Machine Images (AMIs)를 사용하고 인스턴스 실행 시 이러한 AMI를 동적으로 구성할 수 있습니다. AL2023에서도 cloud-init을 사용하여 `ec2-user` 계정을 구성할 수 있습니다.

AL2023에서 `/etc/cloud/cloud.cfg.d`와 `/etc/cloud/cloud.cfg`의 cloud-init을 사용할 수 있습니다. `/etc/cloud/cloud.cfg.d` 디렉터리에 자체 cloud-init 파일을 만들 수 있습니다. Cloud-init

는 이 디렉터리의 모든 파일을 사전순으로 읽습니다. 나중 파일이 이전 파일 값을 덮어씁니다. cloud-init으로 인스턴스를 실행하면 cloud-init 패키지는 다음과 같은 구성 작업을 수행합니다.

- 기본 로케일 설정
- 호스트 이름 설정
- 사용자 데이터 파싱 및 처리
- 호스트 프라이빗 SSH 키 생성
- 간편한 로그인 및 관리를 위해 사용자 공개 SSH 키를 `.ssh/authorized_keys`에 추가
- 패키지 관리를 위한 리포지토리 준비
- 사용자 데이터에 정의된 패키지 작업 처리
- 사용자 데이터에 있는 사용자 스크립트를 실행
- 해당하는 경우 인스턴스 스토어 볼륨 마운트
 - 기본적으로 ephemeral0 인스턴스 스토어 볼륨이 존재하고 유효한 파일 시스템이 있다면 인스턴스 스토어 볼륨은 `/media/ephemeral0`에 마운트됩니다. 그렇지 않다면 마운트되지 않습니다.
 - m1.small 및 c1.medium 인스턴스 유형인 경우 인스턴스와 연결된 모든 스왑 볼륨이 마운트됩니다.
 - 다음 cloud-init 명령을 사용하여 기본 인스턴스 스토어 볼륨 마운트를 재정의할 수 있습니다.

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

마운트 관리에 대한 자세한 내용은 cloud-init 설명서의 [마운트\(Mounts\)](#)를 참조하세요.

- 인스턴스를 실행하면 TRIM을 지원하는 인스턴스 스토어 볼륨은 포맷되지 않습니다. 인스턴스 스토어 볼륨을 마운트하려면 먼저 인스턴스 스토어 볼륨의 파티션을 나누고 포맷해야 합니다.

자세한 내용은 Amazon EC2 사용 설명서의 [인스턴스 스토어 볼륨 TRIM 지원](#) 섹션을 참조하세요.

- 인스턴스 실행 후 disk_setup 모듈을 사용하여 인스턴스 스토어 볼륨의 파티션을 나누고 포맷할 수 있습니다.

자세한 내용은 cloud-init 설명서의 [디스크 설정](#)을 참조하세요.

SELinux에서 cloud-init를 사용하는 방법은 [cloud-init를 사용하여 enforcing을 활성화합니다.](#)를 참조하세요.

cloud-init 사용자 데이터 형식에 대한 자세한 내용은 cloud-init 설명서의 [사용자 데이터 형식](#)을 참조하세요.

보안 업데이트 및 기능

AL2023은 다양한 보안 업데이트 및 솔루션을 제공합니다.

주제

- [관리 업데이트](#)
- [클라우드 보안](#)
- [SELinux 모드](#)
- [규정 준수 프로그램](#)
- [SSH 서버 기본값](#)
- [OpenSSL 3의 주요 기능](#)

관리 업데이트

DNF 및 리포지토리 버전을 사용하여 보안 업데이트를 적용합니다. 자세한 내용은 [AL2023에서 패키지 및 운영 체제 업데이트 관리](#) 섹션을 참조하세요.

클라우드 보안

보안은 AWS와 사용자의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다. 자세한 내용은 [Amazon Linux 2023 보안 및 규정 준수](#) 섹션을 참조하세요.

SELinux 모드

SELinux는 AL2023에서 활성화되고 허용 모드로 기본 설정되어 있습니다. 허용 모드에서는 권한 거부 기록되지만 강제하지는 않습니다.

SELinux 정책은 사용자, 프로세스, 프로그램, 파일 및 장치에 대한 권한을 정의합니다. SELinux에서는 두 정책 중 하나를 선택할 수 있습니다. 정책은 타겟팅 또는 다중 보안 (MLS)입니다.

SELinux 모드 및 정책에 대한 자세한 내용은 [AL2023의 SELinux 모드 설정](#)과 [SELinux Project Wiki](#)를 참조하세요.

규정 준수 프로그램

독립 감사자는 여러 규정 AWS 준수 프로그램과 함께 AL2023 보안 및 규정 준수를 평가합니다.

SSH 서버 기본값

AL2023에 OpenSSH 8.7이 포함되어 있습니다. OpenSSH 8.7은 기본적으로 `ssh-rsa` 키 교환 알고리즘이 비활성화되어 있습니다. 자세한 내용은 [SSH 서버 기본 구성](#) 섹션을 참조하세요.

OpenSSL 3의 주요 기능

- 인증서 관리 프로토콜 (CMP, RFC 4210)에는 CRMF (RFC 4211) 와 HTTP 전송 (RFC 6712) 모두 포함되어 있습니다.
- libcrypto의 HTTP 또는 HTTPS 클라이언트는 GET, POST 작업, 리디렉션, 일반 및 ASN.1으로 인코딩된 콘텐츠, 프록시 및 타임아웃을 지원합니다.
- EVP_KDF는 키 파생 함수와 함께 작동합니다.
- EVP_MAC API는 MACs와 함께 작동합니다.
- 리눅스 커널 TLS 지원.

자세한 내용은 [OpenSSL 마이그레이션 가이드](#)를 참조하세요.

네트워킹 서비스

오픈 소스 프로젝트 `systemd-networkd`는 최신 Linux 배포판에서 널리 사용할 수 있습니다. 이 프로젝트는 나머지 `systemd` 프레임워크와 유사한 선언적 구성 언어를 사용합니다. 기본 구성 파일 유형은 `.network` 및 `.link` 파일입니다.

`amazon-ec2-net-utils` 패키지는 `/run/systemd/network` 디렉터리에 인터페이스별로 구성됩니다. 이렇게 구성하면 인스턴스에 연결된 인터페이스에서 IPv4 네트워킹과 IPv6 네트워킹 모두 사용할 수 있습니다. 또한 로컬에서 소싱된 트래픽이 해당 인스턴스의 네트워크 인터페이스를 통해 네트워크로 라우팅 시키는 정책 라우팅 규칙을 설치합니다. 이러한 규칙은 연결된 주소 또는 접두사에서 탄력적 네트워크 인터페이스(ENI)를 통해 올바른 트래픽이 라우팅되도록 보장합니다. ENI에 대한 자세한 정보는 Amazon EC2 사용 설명서의 [ENI 사용](#) 섹션을 참조하세요.

사용자 지정 구성 파일을 `/etc/systemd/network` 디렉터리에 배치하고 `/run/systemd/network`에 포함된 기본 구성 설정을 재정의하여 이 네트워킹 동작을 사용자 지정할 수 있습니다.

[systemd.network](#) 설명서에 systemd-networkd 서비스가 특정 인터페이스에 적용되는 구성을 결정하는 방법이 나와 있습니다. 또한 ENI 지원 인터페이스가 다양한 AWS 리소스의 속성을 반영할 수 있도록 altnames이라는 대체 이름도 생성합니다. 이러한 ENI 지원 인터페이스 속성은 ENI 첨부 파일의 ENI ID 및 DeviceIndex 필드입니다. ip 명령과 같은 다양한 도구를 사용할 때 해당 속성으로 이러한 인터페이스를 참조할 수 있습니다.

AL2023 인스턴스 인터페이스 이름은 systemd 슬롯 명명 방식으로 생성됩니다. 자세한 내용은 [systemd.net 명명 방식](#)을 참조하세요.

또한 AL2023은 기본적으로 fq_code1 활성 대기열 관리 네트워크 전송 스케줄링 알고리즘을 사용합니다. 자세한 내용은 [CoDel 개요](#)(CoDel overview) 섹션을 참조하세요.

코어 툴체인 패키지 glibc, gcc 및 binutils

Amazon Linux의 일부 패키지는 핵심 툴체인 패키지로 지정됩니다. AL2023의 주요 제품인 Core 패키지는 5년 지원을 받습니다. 패키지 버전은 변경될 수 있지만 Amazon Linux 릴리스에 포함된 패키지는 장기 지원을 받습니다.

이 세 가지 핵심 패키지는 Amazon Linux 배포판에서 대부분의 소프트웨어를 빌드하는 데 사용되는 시스템 도구 체인을 제공합니다.

패키지	정의	용도
glibc 2.34	시스템 C 라이브러리	표준 함수를 제공하는 대부분의 바이너리 프로그램과 프로그램과 커널 간의 인터페이스에 사용할 수 있습니다.
gcc 11.2	gcc 컴파일러 제품군	컴파일 C, C++. Fortran
binutils 2.35	어셈블러, 링커 및 기타 바이너리 도구	바이너리 프로그램을 조작하거나 검사합니다.

glibc 라이브러리 업데이트 후 항상 재부팅하는 것이 좋습니다. 서비스를 제어하는 패키지를 업데이트할 경우 서비스를 다시 시작하여 업데이트를 선택하세요. 시스템을 재부팅하면 이전의 모든 패키지 및 라이브러리가 업데이트됩니다.

패키지 관리 도구

AL2023 기본 소프트웨어 패키지 관리 도구는 DNF입니다. DNF는 AL2 패키지 관리 도구인 YUM의 후속 도구입니다.

DNF는 YUM의 사용법과 비슷합니다. 많은 DNF 명령과 명령 옵션이 YUM 명령과 동일합니다. 명령줄 인터페이스 (CLI) 명령에서 대부분의 경우 `dnf`로 `yum`을 대체할 수 있습니다.

예를 들어, 다음과 같은 AL2 yum 명령인 경우:

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

AL2023에서 이 명령이 다음과 같은 명령이 됩니다.

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

AL2023에서 `yum` 명령을 여전히 사용할 수 있지만 `dnf` 명령에 대한 포인터로 사용할 수 있습니다. 따라서 셸이나 스크립트에서 `yum` 명령을 사용하면 모든 명령과 옵션은 DNF CLI와 동일합니다. YUM CLI와 DNF CLI의 차이점에 대한 자세한 내용은 [YUM와 다른 DNF CLI 변경 사항](#)을 참조하세요.

`dnf` 명령의 명령 및 옵션에 대한 전체 참조는 매뉴얼 페이지 `man dnf`를 참조하세요. 자세한 내용은 [DNF 명령 참조](#) 섹션을 참조하세요.

SSH 서버 기본 구성

몇 년 전의 SSH 클라이언트가 있는 경우 인스턴스에 연결할 때 오류가 발생할 수 있습니다. 일치하는 호스트 키 유형을 찾지 못했다는 오류 메시지가 표시되면 SSH 호스트 키를 업데이트하여 이 문제를 해결하세요.

ssh-rsa 서명 기본 비활성화

AL2023은 레거시 `ssh-rsa` 호스트 키 알고리즘을 비활성화하고 축소된 호스트 키 세트를 생성하는 기본 구성을 포함합니다. 클라이언트는 `ssh-ed25519` 또는 `ecdsa-sha2-nistp256` 호스트 키 알고리즘을 지원합니다.

기본 구성에서는 다음과 같은 키 교환 알고리즘을 모두 사용할 수 있습니다.

- 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

기본적으로 AL2023은 ed25519와 ECDSA 호스트 키를 생성합니다. 클라이언트는 ssh-ed25519 또는 ecdsa-sha2-nistp256 호스트 키 알고리즘을 지원합니다. SSH로 인스턴스에 연결할 때 호환 가능한 알고리즘(예: ssh-ed25519 또는 ecdsa-sha2-nistp256)을 지원하는 클라이언트를 사용해야 합니다. 다른 키 유형을 사용해야 하는 경우 사용자 데이터의 cloud-config 프래그먼트로 생성된 키 목록을 재정의하세요.

다음 예제에서 cloud-config가 ecdsa 및 ed25519 키를 사용하여 rsa 호스트 키를 생성합니다.

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

공개 키 인증에 RSA 키 페어를 사용하는 경우 SSH 클라이언트가 rsa-sha2-256 또는 rsa-sha2-512 서명을 지원해야 합니다. 호환되지 않는 클라이언트 사용 중에 업그레이드할 수 없는 경우 인스턴스에서 ssh-rsa 지원을 다시 활성화하세요. ssh-rsa 지원을 다시 활성화하려면 다음 명령을 사용하여 LEGACY 시스템 암호화 정책을 활성화합니다.

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

호스트 키 관리에 대한 자세한 내용은 [Amazon Linux 호스트 키](#) 섹션을 참조하세요.

AL2023에서 더 이상 사용되지 않는 기능

AL2에서는 더 이상 사용되지 않으며 AL2023에는 존재하지 않는 기능이 여기에 설명되어 있습니다. 이는 AL2에는 있지만 AL2023에는 없는 기능 및 패키지와 같은 기능이며 AL2023에는 추가되지 않습니다. AL2에서 기능이 지원되는 기간에 대한 자세한 내용은 [AL2에서 더 이상 사용되지 않는 기능](#) 섹션을 참조하세요.

AL2023에는 더 이상 사용되지 않는 기능도 있으며 향후 릴리스에서 제거될 예정입니다. 이 장에서는 해당 기능이 무엇인지, 더 이상 지원되지 않는 시기, Amazon Linux에서 제거되는 시기에 대해 설명합니다. 더 이상 사용되지 않는 기능을 이해하면 AL2023을 배포하고 Amazon Linux의 다음 메이저 버전을 준비하는 데 도움이 됩니다.

주제

- [compat- 패키지](#)
- [AL1에서 더 이상 사용되지 않는 기능이 중단되고 AL2에서 제거됨](#)
- [AL2에서는 기능이 더 이상 사용되지 않고 AL2023에서는 제거됨](#)
- [AL2023에서 사용 중단](#)

compat- 패키지

접두사가 compat-인 AL2의 모든 패키지는 패키지의 최신 버전을 위해 아직 다시 빌드되지 않은 이전 바이너리와 바이너리 호환성을 위해 제공됩니다. Amazon Linux의 각 새 메이저 버전은 이전 릴리스의 compat- 패키지를 전달하지 않습니다.

Amazon Linux 릴리스(예: AL2)의 모든 compat- 패키지는 더 이상 사용되지 않으며 후속 버전(예: AL2023)에는 존재하지 않습니다. 업데이트된 라이브러리 버전에 맞게 소프트웨어를 다시 빌드하는 것이 좋습니다.

AL1에서 더 이상 사용되지 않는 기능이 중단되고 AL2에서 제거됨

이 섹션에서는 AL1에서 사용할 수 있고 AL2에서 더 이상 사용할 수 없는 기능에 대해 설명합니다.

Note

AL1의 유지 관리 지원 단계의 일환으로, 일부 패키지 지원 종료(EOL) 날짜가 AL1 EOL 이전으로 되었습니다. 자세한 내용은 [AL1 패키지 지원 설명서](#) 섹션을 참조하세요.

Note

일부 AL1 기능은 이전 릴리스에서 중단되었습니다. 자세한 내용은 [AL1 릴리스 노트](#)를 참조하세요.

주제

- [32비트 x86\(i686\) AMI](#)
- [aws-apitools-*가 AWS CLI로 대체](#)
- [systemd가 AL2에서 upstart를 대체](#)

32비트 x86(i686) AMI

[AL1 2014.09 릴리스](#)의 일환으로, Amazon Linux는 32비트 AMI를 생성하는 마지막 릴리스가 될 것이라고 발표했습니다. 따라서 [AL1 2015.03 릴리스](#)부터 Amazon Linux는 더 이상 32비트 모드 실행을 지원하지 않습니다. AL2는 x86-64 호스트의 32비트 바이너리에 대해 제한된 런타임을 지원했으나 새 32비트 바이너리를 빌드할 수 있는 개발 패키지를 제공하지 않았습니다. AL2023에는 더 이상 32비트 사용자 공간 패키지가 포함되지 않습니다. AL2023으로 마이그레이션하기 전에 64비트 코드로 전환을 완료하는 것이 좋습니다.

AL2023에서 32비트 바이너리를 실행하는 경우, AL2023에서 실행되는 AL2 컨테이너 내에서 AL2의 32비트 사용자 공간을 사용할 수 있습니다.

aws-apitools-*가 AWS CLI로 대체

2013년 9월에 AWS CLI가 릴리스되기 전에 AWS는 사용자가 Amazon EC2 API를 직접 호출할 수 있도록 Java에서 구현된 일련의 명령줄 유틸리티를 제공했습니다. 이러한 도구는 2015년에 중단되었으며 AWS CLI는 명령줄에서 Amazon EC2 API와 상호 작용하는 선호되는 방법이 되었습니다. 명령줄 유틸리티 세트에는 다음 aws-apitools-* 패키지가 포함됩니다.

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

- `aws-apitools-mon`

`aws-apitools-*` 패키지에 대한 업스트림 지원은 2017년 3월에 종료되었습니다. 업스트림 지원이 부족하더라도 Amazon Linux는 사용자에게 이전 버전과의 호환성을 제공하기 위해 `aws-apitools-ec2` 같은 명령줄 유틸리티 중 일부를 계속 제공했습니다. AWS CLI는 적극적으로 유지 관리되므로 `aws-apitools-*` 패키지보다 더 강력하고 완전한 도구이며 모든 AWS API를 사용할 수 있는 수단을 제공합니다.

`aws-apitools-*` 패키지는 2017년 3월에 더 이상 사용되지 않으며 추가 업데이트를 받지 않습니다. 이러한 패키지의 모든 사용자는 가능한 한 빨리 AWS CLI로 마이그레이션해야 합니다. 이러한 패키지는 AL2023에 없습니다.

또한 AL1은 AL1에서 더 이상 사용되지 않으며 AL2부터 Amazon Linux에 없는 `aws-apitools-iam` 및 `aws-apitools-rds` 패키지를 제공했습니다.

systemd가 AL2에서 upstart를 대체

AL2는 systemd init 시스템을 사용하는 첫 번째 Amazon Linux 릴리스로, AL1에서 upstart를 대체했습니다. upstart 특정 구성은 AL1에서 최신 버전의 Amazon Linux로 마이그레이션하는 과정에서 변경해야 합니다. AL1에서는 systemd를 사용할 수 없으므로 upstart에서 systemd로 이동하는 것은 AL2 또는 AL2023과 같은 최신 Amazon Linux 메이저 버전으로 이동하는 과정에서만 수행할 수 있습니다.

AL2에서는 기능이 더 이상 사용되지 않고 AL2023에서는 제거됨

이 섹션에서는 AL2에서 사용할 수 있고 AL2023에서는 더 이상 사용할 수 없는 기능에 대해 설명합니다.

주제

- [32비트 x86\(i686\) 패키지](#)
- [aws-apitools-* 로 대체됨 AWS CLI](#)
- [통합 Amazon CloudWatch Logs 에이전트를 위해 awslogs 사용 중지](#)
- [bzip2 개정 제어 시스템](#)
- [cgroup v1](#)
- [log4j 핫패치\(log4j-cve-2021-44228-hotpatch\)](#)
- [lsb_release 및 system-lsb-core 패키지](#)

- [mccrypt](#)
- [OpenJDK 7\(java-1.7.0-openjdk\)](#)
- [Python 2.7](#)
- [rsyslog-openssl](#)를 대체합니다. [rsyslog-gnutls](#)
- [네트워크 정보 서비스\(NIS\) / yp](#)
- [Amazon VPC create-dhcp-options](#)의 여러 도메인 이름
- [Sun RPC](#)의 [glibc](#)
- [audit 로그의 OpenSSH 키 지문](#)
- [ld.gold 링커](#)
- [ping6](#)
- [ftp 패키지](#)

32비트 x86(i686) 패키지

[AL1 2014.09 릴리스](#)의 일환으로, 32비트 AMI를 생성하는 마지막 릴리스가 될 것이라고 발표했습니다. 따라서 [AL1 2015.03 릴리스](#)부터 Amazon Linux는 더 이상 32비트 모드 실행을 지원하지 않습니다. AL2는 x86-64 호스트의 32비트 바이너리에 대해 제한된 런타임을 지원했으나 새 32비트 바이너리를 빌드할 수 있는 개발 패키지를 제공하지 않았습니다. AL2023에는 더 이상 32비트 사용자 공간 패키지가 포함되지 않습니다. 고객은 64비트 코드로 전환하는 것이 좋습니다.

AL2023에서 32비트 바이너리를 실행하는 경우, AL2023에서 실행되는 AL2 컨테이너 내에서 AL2의 32비트 사용자 공간을 사용할 수 있습니다.

aws-apitools-* 로 대체됨 AWS CLI

2013 AWS CLI 년 9월 릴리스 이전에는에서 구현Java된 일련의 명령줄 유틸리티를 사용할 수 있게 AWS 하여 고객이 Amazon EC2 API를 호출할 수 있게 했습니다. 이러한 도구는 2015년에 더 이상 사용되지 않아 명령줄에서 Amazon EC2 APIs와 상호 작용하는 선호되는 방법이 AWS CLI 되었습니다. 여기에는 다음 aws-apitools-* 패키지가 포함됩니다.

- aws-apitools-as
- aws-apitools-cfn
- aws-apitools-common
- aws-apitools-ec2

- `aws-apitools-elb`
- `aws-apitools-mon`

`aws-apitools-*` 패키지에 대한 업스트림 지원은 2017년 3월에 종료되었습니다. 업스트림 지원이 부족하더라도 Amazon Linux는 고객에게 이전 버전과의 호환성을 제공하기 위해 `aws-apitools-ec2` 같은 명령줄 유틸리티 중 일부를 계속 제공했습니다. AWS CLI는 패키지가 능동적으로 유지 관리되므로 `aws-apitools-*` 패키지보다 더 강력하고 완전한 도구이며 모든 AWS APIs를 사용할 수 있는 수단을 제공합니다.

`aws-apitools-*` 패키지는 2017년 3월에 더 이상 사용되지 않으며 추가 업데이트를 받지 않습니다. 이러한 패키지의 모든 사용자는 가능한 AWS CLI 한 빨리로 마이그레이션해야 합니다. 이러한 패키지는 AL2023에 없습니다.

통합 Amazon CloudWatch Logs 에이전트를 위해 **awslogs** 사용 중지

[awslogs](#) 패키지는 AL2에서 더 이상 사용되지 않으며 AL2023에는 더 이상 존재하지 않습니다. `amazon-cloudwatch-agent` 패키지에서 사용할 수 있는 [통합 CloudWatch Logs 에이전트](#)로 대체됩니다. 자세한 내용은 [Amazon CloudWatch Logs 사용자 안내서](#)를 참조하세요.

bzr 개정 제어 시스템

[GNU Bazaar](#)(bzr) 개정 제어 시스템은 AL2에서 중단되고 AL2023에는 더 이상 존재하지 않습니다.

bzr의 사용자는 리포지토리를 git로 마이그레이션하는 것이 좋습니다.

cgroup v1

AL2023은 통합 제어 그룹 계층 구조(cgroup v2)로 이동하는 반면, AL2는 cgroup v1을 사용합니다. AL2는 cgroup v2를 지원하지 않으므로 AL2023으로 이전하는 과정에서 이 마이그레이션을 완료해야 합니다.

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

Note

`log4j-cve-2021-44228-hotpatch` 패키지는 AL2에서는 더 이상 사용되지 않으며 AL2023에서는 제거됩니다.

[CVE-2021-44228](#)을 위해 AL1 및 AL2용 [Apache Log4j용 핫패치](#) RPM 패키지 버전을 출시했습니다. [Amazon Linux에 핫패치를 추가한다고 발표](#)하면서 "핫패치 설치가 CVE-2021-44228 또는 CVE-2021-45046 완화 기능을 갖춘 log4j 버전 업데이트를 대체하는 것이 아니"라고 언급했습니다.

이 핫패치는 log4j 패치 시간을 벌기 위한 조치였습니다. [CVE-2021-44228](#)에 대응한 지 15개월 후에 첫 AL2023 일반 공급(GA)이 이루어졌기 때문에 AL2023에 (활성화 여부와 관계없이) 핫패치를 포함되어 있지 않습니다.

Amazon Linux에서 자체 log4j 버전을 실행한다면 [CVE-2021-44228](#) 또는 [CVE-2021-45046](#)의 영향을 받지 않는 버전으로 업데이트하는 것이 좋습니다.

lsb_release 및 system-lsb-core 패키지

이전에는 일부 소프트웨어로 lsb_release 명령(AL2에 설치된 system-lsb-core 패키지)을 호출하여 실행 중인 Linux 배포판에 대한 정보를 가져올 수 있었습니다. Linux 표준 베이스(LSB)에서 이 명령을 사용할 수 있고 Linux 배포판에도 이 명령을 설치되었습니다. Linux 배포판은 이 정보를 /etc/os-release 및 기타 관련 파일에 보관하는 더 간단한 표준으로 발전했습니다.

이 os-release 표준은 systemd에서 나왔습니다. 자세한 내용은 [systemd OS 관련 설명서](#)를 참조하세요.

AL2023에 lsb_release 명령이 사용할 수 없으며 system-lsb-core 패키지도 포함되어 있지 않습니다. Amazon Linux 및 기타 주요 Linux 배포판의 호환성을 유지하려면 소프트웨어를 os-release 표준으로 전환해야 합니다.

mcrypt

mcrypt 라이브러리 및 연결된 PHP 확장은 AL2에서 더 이상 사용되지 않으며 AL2023에는 더 이상 존재하지 않습니다.

업스트림 PHP는 [2016년 12월에 처음 릴리스된 PHP 7.1의 mcrypt 확장을 더 이상 사용하지 않고](#) 2019년 10월에 최종 릴리스되었습니다.

업스트림 mcrypt 라이브러리는 [2007년에 마지막으로 릴리스](#)되었으며 [SourceForge가 2017년에 새 커밋에 필요한](#) cvs 개정 제어에서 마이그레이션하지 않았습니다. 가장 최근 커밋(및 3년 전만 해당)은 2011년부터 유지 관리자가 있는 프로젝트에 대한 언급을 제거한 것입니다.

mcrypt의 나머지 사용자는 mcrypt가 AL2023에 추가되지 않으므로 코드를 OpenSSL로 포팅하는 것이 좋습니다.

OpenJDK 7(java-1.7.0-openjdk)

Note

AL2023은 Java 기반 워크로드를 지원하는 다양한 버전의 [Amazon Corretto](#)를 제공합니다. OpenJDK 7 패키지는 AL2에서 더 이상 사용되지 않으며 AL2023에는 더 이상 존재하지 않습니다. AL2023에서 사용할 수 있는 가장 오래된 JDK는 Corretto 8에서 제공합니다.

Amazon Linux Java에 대한 자세한 내용은 [AL2023에 설치된 Java](#) 섹션을 참조하세요.

Python 2.7

Note

AL2023에서 Python 2.7이 제거되었으므로, Python을 사용하는 OS 구성 요소는 Python 3에서 작동됩니다. Amazon Linux가 제공하고 지원하는 Python 버전을 계속 사용하려면 Python 2 코드를 Python 3로 변환하세요.

Amazon Linux Python에 대한 자세한 내용은 [AL2023에서 Python 사용](#) 섹션을 참조하세요.

rsyslog-openssl를 대체합니다. rsyslog-gnutls

rsyslog-gnutls 패키지는 AL2에서는 더 이상 사용되지 않으며 AL2023에는 더 이상 존재하지 않습니다. rsyslog-openssl 패키지는 패키지 사용에 대한 드롭인 대체 rsyslog-gnutls 패키지여야 합니다.

네트워크 정보 서비스(NIS) / yp

원래 노란색 페이지 또는 YP라고 하는 네트워크 정보 서비스(NIS)는 AL2에서 더 이상 사용되지 않으며 AL2023에는 더 이상 존재하지 않습니다. 여기에는 ypbind, ypserv 및 yp-tools 패키지가 포함됩니다. NIS와 통합되는 다른 패키지는 AL2023에서 이 기능이 제거되었습니다.

Amazon VPC create-dhcp-options의 여러 도메인 이름

Amazon Linux 2에서는 domain-name 파라미터의 여러 도메인 이름을 [create-dhcp-options](#)에 전달하여 /etc/resolv.conf에 search foo.example.com bar.example.com 같은 항목이 포

함될 수 있습니다. Amazon VPC DHCP 서버는 단일 도메인 이름만 지원하는 DHCP 옵션 15를 사용하여 제공된 도메인 이름 목록을 전송합니다([RFC 2132 섹션 3.17](#) 참조). AL2023은 RFC를 따르는 네트워크 구성에 systemd-networkd를 사용하므로 AL2의 이 우발적 기능은 AL2023에 없습니다.

[AWS CLI](#) 및 [Amazon VPC 설명서](#)에는 다음의 문구가 있습니다. "일부 Linux 운영 체제는 공백으로 구분된 여러 도메인 이름을 허용합니다. 하지만 Windows와 기타 Linux 운영 체제에서는 이 값을 단일 도메인으로 취급하므로 예기치 않은 동작이 발생합니다. DHCP 옵션 세트가 값을 단일 도메인으로 취급하는 운영 체제가 실행되는 인스턴스가 있는 VPC와 연결되어 있는 경우 도메인 이름을 하나만 지정합니다."

AL2023과 같은 이러한 시스템에서 DHCP 옵션 15(하나만 허용)를 사용하여 두 개의 도메인을 지정하면 [도메인 이름에 공백 문자가 유효하지 않으므로](#) 공백 문자가 032로 인코딩되어 /etc/resolv.conf가 search foo.exmple.com032bar.example.com을 포함하게 됩니다.

여러 도메인 이름을 지원하려면 DHCP 서버가 DHCP 옵션 119를 사용해야 합니다([RFC 3397, 섹션 2](#) 참조). Amazon VPC DHCP 서버에서 지원되는 경우 [Amazon VPC 사용 설명서](#)를 참조하세요.

Sun RPC의 glibc

Sun RPC에서의 glibc 구현은 AL2에서는 더 이상 사용되지 않으며 AL2023에서는 제거됩니다. Sun RPC 기능이 필요한 경우 libtirpc 라이브러리(AL2 및 AL2023에서 사용 가능)를 사용하여 이동하는 것이 좋습니다. 또한 libtirpc를 채택하면 애플리케이션이 IPv6을 지원할 수 있습니다.

이 변경 사항은 [Fedora의 glibc에서 Sun RPC 인터페이스 제거](#) 및 [Gentoo의 유사한 변경](#) 등 이러한 기능을 제거하는 glibc 업스트림의 광범위한 커뮤니티 채택을 반영합니다.

audit 로그의 OpenSSH 키 지문

AL2 수명 주기 후반부에 패치가 OpenSSH 패키지에 추가되어 인증에 사용되는 키 지문을 내보냈습니다. 이 기능은 AL2023에는 없습니다.

ld.gold 링커

ld.gold 링커는 AL2에서 사용할 수 있으며 AL2023에서 제거됩니다. gold 링커를 명시적으로 참조하는 소프트웨어를 빌드하는 고객은 일반 (ld.bfd) 링커로 마이그레이션해야 합니다.

업스트림 [GNU Binutils 버전 2.44\(2025년 2월 릴리스\)의 릴리스 노트](#)에는 다음과 같은 ld.gold에 대한 제거 사항이 명시되어 있습니다: "이전 관행과 달리, 이번 릴리스에서 binutils-2.44.tar tarball에는 gold 링커의 소스가 포함되어 있지 않습니다. 이는 이제 골드 링커가 더 이상 사용되지 않으며, 지원자가 앞으로 나아가서 개발 및 유지 관리를 계속할 것을 제안하지 않는 한 결국 제거되기 때문입니다."

ping6

AL2023에서는 일반 ping 유틸리티가 기본적으로 IPv6를 지원하므로 별도의 /bin/ping6이 더 이상 필요하지 않습니다. AL2023에서 /usr/sbin/ping6은 /usr/bin/ping 실행 파일에 대한 symlink입니다.

이 변경은 [Fedora의 Ping IPv6 변경](#) 등 이러한 기능을 제공하는 더 광범위한 커뮤니티의 최신 iputils 버전 채택을 따릅니다.

ftp 패키지

AL2의 ftp 패키지는 AL2023부터 Amazon Linux에서 더 이상 사용할 수 없습니다. 이 결정은 보안, 유지 관리 가능성 및 최신 소프트웨어 개발 관행에 대한 지속적인 노력의 일환으로 이루어졌습니다. AL2023 마이그레이션의 일환으로(또는 마이그레이션 이전에) 레거시 ftp 패키지 사용을 대체 패키지 중 하나로 마이그레이션하는 것이 좋습니다.

배경

레거시 ftp 패키지는 수년 동안 업스트림에서 적극적으로 유지되지 않았습니다. 소스 코드에 대한 마지막 중요 업데이트는 2000년대 초에 이뤄졌으며 원래 소스 리포지토리를 더 이상 사용할 수 없습니다. 일부 Linux 배포판에는 보안 취약성에 대한 패치가 포함되어 있지만 코드베이스는 대부분 유지 관리되지 않습니다.

권장 대안

AL2023은 FTP 기능에 대한 몇 가지 현대적이고 적극적으로 유지 관리되는 대안을 제공합니다.

lftp(AL2 및 AL2023에서 사용 가능)

FTP, HTTP, SFTP 및 기타 프로토콜을 지원하는 정교한 파일 전송 프로그램입니다. 기존 ftp 클라이언트보다 더 많은 기능을 제공하며 적극적으로 유지 관리됩니다.

다음을 사용하여 설치: `dnf install lftp`

curl(AL2 및 AL2023에서 사용 가능)

URL, FTPS, HTTP, HTTPS 및 기타 여러 프로토콜을 사용하여 데이터를 전송하기 위한 다목적 명령줄 도구입니다.

curl-minimal 패키지를 통해 AL2023에서 기본적으로 사용할 수 있습니다. 보다 광범위한 프로토콜 지원을 위해 선택적으로 `dnf swap curl-minimal curl-full`을 사용하여 curl-full로 업그레이드할 수 있습니다.

wget(AL2 및 AL2023에서 사용 가능)

웹에서 파일을 다운로드하고 HTTP, HTTPS 및 FTP 프로토콜을 지원하는 비대화형 명령줄 유틸리티입니다.

다음을 사용하여 설치: `dnf install wget`(일부 AL2023 이미지에는 기본적으로 설치되지 않음)

sftp(AL2 및 AL2023에서 사용 가능)

SSH를 통해 작동하는 보안 파일 전송 프로토콜로, 암호화된 파일 전송을 제공합니다.

기본적으로 OpenSSH 패키지의 일부로 사용할 수 있습니다.

마이그레이션 고려 사항

애플리케이션 또는 스크립트가 레거시 ftp 클라이언트에 종속되는 경우 다음 마이그레이션 접근 방식을 고려하세요.

1. 최신 대안을 사용하도록 스크립트 업데이트: 레거시 ftp 클라이언트 대신 `lftp`, `curl`, `wget` 또는 `sftp`를 사용하도록 스크립트를 수정합니다.
2. 패키지 종속성 검토: 일부 애플리케이션은 내부적으로 최신 프로토콜을 사용하여 ftp로 마이그레이션한 지 오래 되었지만 패키지 메타데이터에 패키지를 종속성으로 나열할 수 있습니다. 이러한 경우 ftp 패키지에 `/usr/bin/ftp`가 없더라도 애플리케이션이 AL2023에서 올바르게 작동할 수 있습니다. 명시된 종속성에만 의존하지 않고 애플리케이션의 실제 요구 사항을 검토합니다.
3. 애플리케이션 종속성 업데이트: ftp 패키지에 대한 종속성을 여전히 선언하지만 실제로 사용하지 않는 애플리케이션을 유지 관리하는 경우 패키지 메타데이터를 업데이트하여 불필요한 종속성을 제거합니다.

보안 고려 사항

FTP 프로토콜은 인증 자격 증명을 포함한 데이터를 일반 텍스트로 전송합니다. 보안에 민감한 애플리케이션의 경우 권장 대체 도구에서 지원하는 SFTP 또는 HTTPS와 같은 암호화된 대안을 사용하는 것이 좋습니다.

AL2023에서 사용 중단

이 섹션에서는 AL2023에 존재하며 향후 버전의 Amazon Linux에서 제거될 가능성이 높은 기능에 대해 설명합니다. 각 섹션에서는 기능이 무엇이고 Amazon Linux에서 언제 제거될 것으로 예상되는지 설명합니다.

Note

이 섹션은 Linux 에코시스템이 발전하고 향후 Amazon Linux의 메이저 버전이 릴리스에 가까워짐에 따라 차차 업데이트됩니다.

주제

- [32비트 x86\(i686\) 런타임 지원](#)
- [aspell](#)
- [Berkeley DB\(libdb\)](#)
- [cron](#)
- [IMDSv1](#)
- [pcre 버전 1](#)
- [System V init \(sysvinit\)](#)
- [EOL 패키지는 더 이상 사용되지 않습니다.](#)

32비트 x86(i686) 런타임 지원

AL2023은 32비트 x86(i686) 바이너리를 실행할 수 있는 기능을 유지합니다. Amazon Linux의 다음 메이저 버전은 더 이상 32비트 사용자 공간 바이너리 실행을 지원하지 않을 수 있습니다.

aspell

AL2023은 aspell 패키지와 함께 제공되지만 더 이상 사용되지 않으며 Amazon Linux의 다음 메이저 릴리스에서 제거될 예정입니다. 고객은 hunspell 또는 enchant2 등 최신 교체로 마이그레이션하는 것이 좋습니다.

AL2023에서 aspell의 사용 중단은 [Fedora에서의 aspell 사용 중단](#)과 같은 광범위한 커뮤니티 전환을 따릅니다.

Berkeley DB(**libdb**)

AL2023은 Berkeley DB(libdb) 라이브러리 버전 5.3.28과 함께 제공됩니다. 덜 제한적인 Sleepycat 라이선스에서 라이선스가 GNU Affero GPLv3(AGPL) 라이선스로 변경되기 전의 Berkeley DB의 마지막 버전입니다.

AL2023에는 Berkeley DB(libdb)에 의존하는 패키지가 거의 없으며, 라이브러리는 Amazon Linux의 다음 메이저 릴리스에서 제거될 예정입니다.

Note

AL2023의 dnf 패키지 관리자는 Berkeley DB(BDB) 형식 rpm 데이터베이스에 대한 읽기 전용 지원을 유지합니다. 이 지원은 Amazon Linux의 다음 메이저 릴리스에서 제거됩니다.

libdb의 사용 중단은 [Fedora에서의 libdb 사용 중단](#)과 같은 광범위한 커뮤니티 전환을 따릅니다.

cron

cronie 패키지는 AL2 AMI에 기본적으로 설치되었으며, 정기 작업을 예약하는 기존 crontab 방식을 지원합니다. AL2023에서는 cronie이 기본적으로 포함되지 않습니다. 따라서 crontab에 대한 지원은 더 이상 기본적으로 제공되지 않습니다.

AL2023에서는 cronie 패키지를 옵션으로 설치하여 기존 cron 작업을 할 수 있습니다. systemd에서 추가 기능을 제공하므로 systemd 타이머로 마이그레이션하는 것이 좋습니다.

향후 버전의 Amazon Linux, 아마도 다음 메이저 버전에는 더 이상 클래식 cron 작업에 대한 지원이 포함되지 않고 systemd 타이머로의 전환을 완료할 수 있습니다. cron을 사용하여 마이그레이션하는 것이 좋습니다.

IMDSv1

기본적으로 AL2023 AMI는 IMDSv2 전용 모드에서 시작되도록 구성되어 IMDSv1의 사용을 비활성화합니다. IMDSv1이 활성화된 상태에서 AL2023을 사용할 수 있는 옵션은 여전히 있습니다. Amazon Linux의 향후 버전은 IMDSv2만 적용할 가능성이 높습니다.

AMI의 IMDS 구성에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [AMI 구성](#) 섹션을 참조하세요.

pcre 버전 1

레거시 pcre 패키지는 더 이상 사용되지 않으며 Amazon Linux의 다음 메이저 릴리스에서 제거될 예정입니다. pcre2 패키지의 후속 버전입니다. AL2023의 첫 번째 버전은 pcre에 대해 빌드되는 제한된 수의 패키지와 함께 제공되었지만 이러한 패키지는 AL2023 내에서 pcre2로 마이그레이션됩니다. 더 이상 사용되지 않는 pcre 라이브러리는 AL2023에서 계속 사용할 수 있습니다.

Note

pcr의 더 이상 사용되지 않는 버전은 AL2023의 전체 수명 동안 보안 업데이트를 받지 않습니다. pcr 지원 수명 주기 및 패키지가 보안 업데이트를 수신하는 시간에 대한 자세한 내용은 [pcr 패키지의 패키지 지원 설명](#)을 참조하세요.

pcr2에 유리한 pcr의 사용 중단은 [Fedora에서의 pcr 사용 중단](#)과 같이 이러한 방향의 더 광범위한 커뮤니티 전환을 따릅니다.

System V init (sysvinit)

AL2023은 System V 서비스(init) 스크립트와 이전 버전과의 호환성을 유지하지만 업스트림 systemd 프로젝트는 [v254 릴리스](#)의 일부로 [System V 서비스 스크립트에 대한 지원 중단](#)을 발표하고 향후 systemd 버전에서 지원이 제거될 것이라고 명시했습니다. [자세한 내용은 systemd 단원을 참조하세요.](#)

AL2023은 System V 서비스(init) 스크립트와 이전 버전과의 호환성을 유지하지만 사용자는 다음 메이저 릴리스에서 System V 서비스(init) 스크립트에 대한 지원이 Amazon Linux에서 제거될 때를 대비하기 위해 네이티브 systemd 단위 파일을 사용하여 마이그레이션하는 것이 좋습니다.

EOL 패키지는 더 이상 사용되지 않습니다.

AL2023에서 사용할 수 있는 각 패키지에는 Amazon Linux 관련 정보를 다루는 관련 [지원 설명서](#)가 있습니다. 이러한 문은 OS의 핵심 및 수명과 [the section called “PHP”](#) 및 [the section called “Python”](#)과 같은 패키지를 다룹니다. 여기서 AL2023은 여러 버전을 제공하며 각 버전은 업스트림 오픈 소스 프로젝트가 수행하는 기간 동안 지원됩니다.

AL2023에서는 dnf 패키지 관리자를 사용하여 패키지 지원 정보를 가져올 수 있습니다. 자세한 내용은 [패키지 지원 정보 가져오기](#) 섹션을 참조하세요.

Amazon Linux의 메이저 버전이 종료되기 전에 패키지가 더 이상 지원되지 않는 경우 이 패키지는 더 이상 사용되지 않으며 Amazon Linux의 다음 메이저 버전에 존재하지 않을 것으로 가정해야 합니다.

각 메이저 Amazon Linux 버전이 각각 지원 수명 주기가 다른 여러 버전을 배송한 [the section called “PHP”](#) 및 [the section called “Python”](#)과 같은 패키지의 경우 패키지의 메이저 버전이 거의 또는 전혀 중복되지 않지만 Amazon Linux의 새 메이저 버전에 계속 존재할 가능성이 높습니다. 종속성을 선택할 때는 Amazon Linux 패키지 지원 타임라인을 염두에 두는 것이 좋습니다.

AL2와 AL2023 비교

다음 주제에서는 AL2와 AL2023의 주요 차이점을 설명합니다.

AL1, AL2 및 AL2023에서 더 이상 사용되지 않는 기능에 대한 자세한 내용은 [AL2023에서 더 이상 사용되지 않는 기능](#) 섹션을 참조하세요.

주제

- [패키지 추가, 업그레이드 및 삭제](#)
- [각 릴리스에 대한 지원](#)
- [이름 지정 및 버전 변경](#)
- [최적화](#)
- [다수 업스트림 제공](#)
- [네트워킹 시스템 서비스](#)
- [패키지 관리자](#)
- [cloud-init 사용하기](#)
- [데스크톱 그래픽 지원](#)
- [컴파일러 트리플렛](#)
- [32비트 x86 \(i686\) 패키지](#)
- [lsb_release 및 system-lsb-core 패키지](#)
- [Extra Packages for Enterprise Linux \(EPEL\)](#)
- [Python 2.7은 Python 3으로 대체되었습니다.](#)
- [보안 업데이트](#)
- [안정성을 위한 결정적 업그레이드](#)
- [기본 Amazon EBS 볼륨 유형인 gp3](#)
- [통합 제어 그룹 계층 구조 \(cgroup v2\)](#)
- [systemd 타이머 교체 cron](#)
- [향상된 도구 체인: gcc, binutils 및 glibc](#)
- [systemd 저널 대체 rsyslog](#)
- [패키지 종속성 최소화](#)
- [기본 JVM으로 Amazon Corretto 설정](#)
- [AWS CLIv2](#)

- [UEFI 기본 설정 및 보안 부팅](#)
- [SSH 서버 기본 구성 변경](#)
- [AL2 이후 AL2023 커널 변경 사항](#)
- [/tmp가 이제 tmpfs입니다.](#)
- [AMI 및 컨테이너 이미지 변경](#)
- [Amazon Linux 2와 Amazon Linux 2023 AMI에 설치된 패키지 비교](#)
- [Amazon Linux 2 및 Amazon Linux 2023 미니멀 AMI에 설치된 패키지 비교](#)
- [Amazon Linux 2 및 Amazon Linux 2023 기본 컨테이너 이미지에 설치된 패키지 비교](#)

패키지 추가, 업그레이드 및 삭제

AL2023에는 사용할 수 있는 수천 개의 소프트웨어 패키지가 포함되어 있습니다. Amazon Linux 구 버전과 비교해서 AL2023 버전에 추가, 업그레이드 또는 삭제된 모든 패키지의 전체 목록은 [AL2023 패키지 변경 사항](#)을 참조하세요.

AL2023에 패키지 추가 또는 변경 요청을 하려면 GitHub의 [amazon-linux-2023 리포지토리](#)에 이슈를 제출합니다.

각 릴리스에 대한 지원

AL2023은 5년간 지원합니다.

자세한 내용은 [릴리스 케이던스](#) 단원을 참조하십시오.

이름 지정 및 버전 변경

AL2023은 플랫폼 식별을 위해 AL2가 지원하는 것과 동일한 메커니즘을 제공합니다. 또한 AL2023에 플랫폼 식별을 위한 새 파일을 제공합니다.

자세한 내용은 [이름 지정 및 버전 관리](#) 단원을 참조하십시오.

최적화

AL2023 부팅 시간을 최적화하여 인스턴스 실행부터 고객 워크로드 실행까지의 시간이 단축됩니다. Amazon EC2 인스턴스 커널 구성, cloud-init 구성 및 kmod와 systemd 같은 OS 패키지에 내장된 기능을 최적화합니다.

최적화에 대한 자세한 내용은 [성능 및 운영 최적화](#)를 참조하세요.

다수 업스트림 제공

AL2023 버전은 RPM 기반이며 여러 버전의 Fedora 및 기타 배포판 (예: CentOS 9 Stream) 에서 가져온 구성 요소를 포함합니다. Amazon Linux 커널은 kernel.org에서 직접 제공하는 장기 지원 (LTS) 릴리스에서 가져온 것으로, 다른 배포판과는 별개로 선택된 것입니다.

자세한 내용은 [Fedora와의 관계](#) 단원을 참조하십시오.

네트워킹 시스템 서비스

systemd-networkd 시스템 서비스는 AL2023 네트워크 인터페이스를 관리합니다. 이는 ISC dhclient 또는 dhclient를 사용하는 AL2와 다른 점입니다.

자세한 내용은 [네트워킹 서비스](#) 단원을 참조하십시오.

패키지 관리자

AL2023 기본 소프트웨어 패키지 관리 도구는 DNF입니다. DNF는 AL2 패키지 관리 도구인 YUM의 후속 도구입니다.

자세한 내용은 [패키지 관리 도구](#) 단원을 참조하십시오.

cloud-init 사용하기

cloud-init는 AL2023 패키지 리포지토리를 관리합니다. Amazon Linux 구 버전 cloud-init에 기본적으로 보안 업데이트가 설치되어 있습니다. 이는 AL2023 기본값이 아닙니다. AL2023에서 실행할 때 releasever 업데이트하는 새로운 결정적 업그레이드 기능은 실행할 때 패키지 업데이트를 활성화할 수 있습니다. 자세한 내용은 [AL2023에서 패키지 및 운영 체제 업데이트 관리](#) 및 [안정성을 위한 결정적 업그레이드](#) 섹션을 참조하세요.

AL2023에서 cloud-init와 SELinux를 함께 사용할 수 있습니다. 자세한 내용은 [cloud-init를 사용하여 enforcing을 활성화합니다](#) 단원을 참조하십시오.

Cloud-init은 HTTP(S)를 사용하여 원격 위치에서 cloud-init으로 구성 내용을 로드합니다. Amazon Linux 구 버전에서는 원격 리소스를 사용할 수 없을 때 알림을 보내지 않았습니다. AL2023 환경에서 원

격 리소스를 사용할 수 없으면 치명적인 오류가 발생하여 cloud-init 실행을 할 수 없습니다. AL2과 다른 이 동작으로 더 안전하게 “실패 종료”를 할 수 있습니다.

자세한 내용은 [cloud-init 설명서](#)의 [사용자 지정 cloud-init](#) 섹션을 참조하세요.

데스크톱 그래픽 지원

AL2023에는 릴리스 2023.7부터 GNOME 기반 그래픽 데스크톱 환경이 탑재되어 AL2에 사용되는 MATE 데스크톱을 대체합니다. 이 버전은 AL2023의 클라우드 최적화 성능을 유지하면서 사용자에게 다른 데스크톱 환경을 제공합니다. GNOME 데스크톱 환경은 다양한 사용자 지정 옵션, 시스템 통합 기능 및 고유한 사용자 인터페이스 설계를 제공하여 사용자에게 이전 MATE 데스크톱 환경의 대안을 제공합니다. 자세한 내용은 [AL2023 그래픽 데스크톱](#) 페이지를 참조하세요.

컴파일러 트리플렛

AL2023에서 컴파일러 트리플렛을 GCC와 LLVM로 설정하면 amazon이 공급업체임을 나타냅니다.

따라서 AL2 aarch64-redhat-linux-gcc는 AL2023에서 aarch64-amazon-linux-gcc가 됩니다.

이는 대부분의 사용자에게 완전히 투명하게 공개되어야 하며, AL2023에서 컴파일러를 빌드하는 사용자에게만 영향을 미칠 수 있습니다.

32비트 x86 (i686) 패키지

[AL1 2014.09 릴리스](#)로 32비트 AMI를 생성하는 마지막 릴리스가 될 것이라고 발표했습니다. 따라서 [AL1 2015.03 릴리스](#)부터 Amazon Linux는 더 이상 32비트 모드에서 실행되지 않습니다. AL2는 x86-64 호스트의 32비트 바이너리에 대해 제한된 런타임을 지원했으나 새 32비트 바이너리를 빌드할 수 있는 개발 패키지를 제공하지 않았습니다. AL2023에는 더 이상 32비트 사용자 공간 패키지가 없습니다. 64 비트 코드로 전환을 완료하는 것이 좋습니다.

AL2023에서 32비트 바이너리를 실행하는 경우, AL2023에서 실행되는 AL2 컨테이너 내에서 AL2의 32 비트 사용자 공간을 사용할 수 있습니다.

lsb_release 및 system-lsb-core 패키지

이전에는 일부 소프트웨어로 lsb_release 명령(AL2에 설치된 system-lsb-core 패키지)을 호출하여 실행 중인 Linux 배포판에 대한 정보를 가져올 수 있었습니다. Linux 표준 베이스(LSB)에서 이 명

령을 사용할 수 있고 Linux 배포판에도 이 명령을 설치되었습니다. Linux 배포판은 이 정보를 `/etc/os-release` 및 기타 관련 파일에 보관하는 더 간단한 표준으로 발전했습니다.

이 `os-release` 표준은 `systemd`에서 나왔습니다. 자세한 내용은 [systemd OS 관련 설명서](#)를 참조하세요.

AL2023에 `lsb_release` 명령이 사용할 수 없으며 `system-lsb-core` 패키지도 포함되어 있지 않습니다. Amazon Linux 및 기타 주요 Linux 배포판의 호환성을 유지하려면 소프트웨어를 `os-release` 표준으로 전환해야 합니다.

Extra Packages for Enterprise Linux (EPEL)

Warning

AL2 `epel` Extra는 타사 EPEL7 리포지토리를 활성화했습니다. 2024년 6월 30일부터 타사 EPEL7 리포지토리는 더 이상 유지 관리되지 않습니다.

이 타사 리포지토리는 향후 업데이트되지 않습니다. 즉, EPEL 리포지토리의 패키지에 대한 보안 수정 사항이 없습니다.

이 섹션에서는에 있는 패키지에 대한 AL2023의 옵션을 다룹니다EPEL.

Extra Packages for Enterprise Linux (EPEL)는 엔터프라이즈급 Linux 운영 체제를 위한 다양한 패키지를 만드는 것을 목표로 하는 Fedora 커뮤니티의 프로젝트입니다. 이 프로젝트를 통해 주로 RHEL 및 CentOS 패키지를 제작했습니다. AL2는 높은 수준의 CentOS 7 호환성을 제공합니다. 결과적으로 많은 EPEL7 패키지가 AL2에서 작동합니다.

AL2023과 호환되는 바이너리 EPEL 버전은 없습니다. 그러나 AL2023에서 EPEL7 패키지를 사용하려는 고객에게는 몇 가지 옵션이 있습니다. 일부 EPEL 패키지에는 AL2023의 대안이 있는 반면, 다른 패키지는의 일부로 제공됩니다[Amazon Linux용 보조 패키지](#).

Warning

AL2023과 함께 사용하도록 설계된 리포지토리만 추가합니다.

다른 배포용으로 설계된 리포지토리는 현재 작동할 수 있지만 AL2023의 패키지 업데이트 또는 AL2023과 함께 사용하도록 설계되지 않은 리포지토리에서도 계속 작동할 것이라는 보장은 없습니다.

이 페이지에서는 AL2 및 해당 AL2023에서 고객이 사용하는 EPEL7 패키지에 대한 정보를 제공합니다.

나머지 패키지의 경우 고객은 Amazon Linux용 보조 패키지(SPAL)를 사용할 수 있습니다. SPAL은 Amazon Linux 2023용으로 특별히 구축된 수천 개의 EPEL9 패키지를 제공하지만 이러한 패키지는 AWS지원 플랜에서 다루지 않습니다. 즉, CVEs SPAL 패키지에 대해 추적되지 않으며 패치는 업스트림을 사용할 수 있는 경우에만 제공됩니다.

 Important

사용하기 [Amazon Linux용 보조 패키지](#) 전의 설명서를 참조하세요.

주제

- [axel - HTTP/FTP 클라이언트](#)
- [brotli 및 libbrotli - 압축](#)
- [collectd - 통계 수집 대몬](#)
- [cpulimit - CPU 사용량 제한기](#)
- [exim - 우편 전송 에이전트](#)
- [fuse3 - 사용자 공간의 파일 시스템\(FUSE\) v3](#)
- [ganglia - 분산 모니터링 시스템](#)
- [git-lfs - Git을 사용하여 대용량 파일 버전 관리](#)
- [haveged - HAVEGE 알고리즘을 사용하는 엔트로피 소스](#)
- [inotify-tools - inotify 명령줄 도구](#)
- [iperf - TCP/UDP 성능 벤치마크](#)
- [jemalloc - 대체 malloc 구현](#)
- [libbsd - BSD 호환 함수 라이브러리](#)
- [libserf - HTTPS 클라이언트 라이브러리](#)
- [libzstd - zstd 압축 라이브러리](#)
- [lighttpd 웹 서버](#)
- [lshell - 제한된 셸](#)
- [monit - 프로세스, 파일, 디렉터리 및 디바이스 모니터](#)
- [nodejs](#)
- [perl-Config-General](#)
- [python2-lockfile - 파일 잠금](#)

- [python2-rsa](#) - 순수 Python RSA
- [python2-simplejson](#) - Python 2에 대한 JSON 루틴
- [rkhunter](#) - Rootkit Hunter
- [rssh](#) - OpenSSH와 함께 사용할 수 있는 제한된 셸
- [sscg](#) - 자체 서명된 SSL 인증서 생성기
- [stress](#) - 스트레스 테스트
- [stress-ng](#) - 스트레스 테스트
- [tmpwatch](#) - 마지막으로 액세스한 시간을 기준으로 파일 제거
- [xmlstarlet](#) - 명령줄 XML 유틸리티

axel - HTTP/FTP 클라이언트

axel 패키지는 EPEL7에 있으며 Amazon Linux의 일부로 배송된 적이 없습니다. AL2023에서 사용할 수 있는 대안은 curl 및 wget입니다.

Warning

axel의 -S 옵션은 암호화되지 않은 http 연결을 사용하여 파일의 미러를 검색합니다.

axel을 사용하여 curl 또는 wget으로 마이그레이션하는 것이 좋습니다.

brotli 및 libbrotli - 압축

brotli 및 libbrotli 패키지는 EPEL7에 있었지만 brotli 패키지만 AL2 Core에서 사용할 수 있었습니다.

brotli 및 libbrotli 패키지는 모두 AL2023에 포함되어 있습니다.

다음 명령을 사용하여 AL2023에 brotli 패키지를 설치할 수 있습니다.

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

다음 명령을 사용하여 AL2023에 libbrotli 패키지를 설치할 수 있습니다.

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

collectd - 통계 수집 대몬

collect 패키지는 EPEL7에 있으며 collectd 및 collectd-python3 AL2 Extra에서도 사용할 수 있습니다.

collectd 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

cpulimit - CPU 사용량 제한기

Amazon Linux 2023에서 systemd는 프로세스 또는 프로세스 그룹의 CPU 사용을 제한하는 기능을 제공합니다. 또한 이 기능은 모든 systemd 서비스에 쉽게 사용할 수 있습니다.

systemd에서 제공하는 강력한 리소스 제어 기능은 태스크 또는 태스크 그룹이 사용할 수 있는 리소스에서 제한되도록 하는 데 사용할 수 있습니다. 자세한 내용은 [systemd를 사용하여 AL2023에서 프로세스 리소스 사용 제한](#) 섹션과 함께 업스트림 [systemd.resource-control](#) 설명서를 참조하세요.

exim - 우편 전송 에이전트

exim 패키지는 EPEL7에 있으며 이전에는 AL1에서 사용할 수 있었습니다. Amazon Linux 2023은 postfix 및 sendmail 우편 전송 에이전트(MTA)를 모두 제공합니다.

fuse3 - 사용자 공간의 파일 시스템(FUSE) v3

fuse3 패키지(fuse3-libs 및 fuse3-devel 포함)는 EPEL7에 있었습니다. 이러한 패키지는 AL2023의 일부이며 다음 명령을 실행하여 각각 설치할 수 있습니다.

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

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

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

ganglia - 분산 모니터링 시스템

ganglia 패키지는 EPEL7에 있으며 이전에는 AL1에서 사용할 수 있었습니다. AL2와 함께 배송되지 않았습니다.

업스트림 프로젝트에는 일부 열린 CVE가 해결되지 않는 비활성 기간이 있었습니다. 업스트림 프로젝트에서 최근 활동이 있었지만 AL2023에 ganglia를 추가할 계획은 없습니다.

git-lfs - Git을 사용하여 대용량 파일 버전 관리

git-lfs 패키지는 EPEL7에 있었습니다. Amazon Linux 2023에서는 git-lfs 패키지가 Core 리포지토리에 포함됩니다. AL2023에서 다음 명령을 실행하여 git-lfs를 설치할 수 있습니다.

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

haveged - HAVEGE 알고리즘을 사용하는 엔트로피 소스

haveged 패키지는 EPEL7에 있었습니다. Amazon Linux 2023은 엔트로피 소스로 사전 구성되어 있으므로 haveged를 사용할 필요가 없습니다.

inotify-tools - inotify 명령줄 도구

inotify-tools 패키지는 EPEL7에 있으며 AL2023에 포함되어 있습니다.

Note

AL2023에서 systemd는 경로가 존재하거나 변경되는 경우 같은 이벤트에 대한 작업을 수행하는 데 사용할 수 있는 경로 기반 활성화를 지원합니다.

사용되는 대부분의 inotify-tools를 이제 systemd 경로 활성화로 보다 안정적인 방식으로 더 잘 수행할 수 있습니다. 자세한 내용은 [systemd.path](#) 섹션을 참조하세요.

inotify-tools 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

iperf - TCP/UDP 성능 벤치마크

iperf 버전 2 패키지는 EPEL7에 있으며 testing AL2 Extra에서도 사용할 수 있습니다. AL1에서도 사용할 수 있습니다.

Note

iperf3 패키지도 사용할 수 있으며 iperf 버전 3을 제공합니다.

iperf 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

jemalloc - 대체 malloc 구현

jemalloc 패키지는 EPEL7에 있으며 lamp-mariadb10.2-php7.2 및 mariadb10.5 AL2 Extra에서도 사용할 수 있습니다.

jemalloc 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

libbsd - BSD 호환 함수 라이브러리

libbsd 패키지는 EPEL7에 있으며 testing AL2 Extra에서도 사용할 수 있습니다.

libbsd 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

다음 명령을 실행하여 libbsd에 대한 개발 파일을 설치할 수 있습니다.

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

libserf - HTTPS 클라이언트 라이브러리

libserf 패키지는 EPEL7에 있었습니다. libserf 패키지는 Amazon Linux 2023에서 제공됩니다. 다음 명령을 실행하여 설치할 수 있습니다.

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

libzstd - zstd 압축 라이브러리

libzstd 패키지는 AL2 Core와 EPEL7에 있었습니다. libzstd 패키지는 AL2023의 일부이기도 합니다.

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


lighttpd 웹 서버

lighttpd 패키지는 EPEL7에 있으며 이전에는 AL1에서 사용할 수 있었습니다. Amazon Linux 2023은 Apache httpd 및 nginx 웹 서버를 모두 제공합니다.

lshe11 - 제한된 셸

이 lshe11 패키지는 Amazon Linux의 일부로 배송된 적이 없습니다. EPEL6에서 사용할 수 있었습니다. [lshe11용 Fedora 패키징 리포지토리](#)는 EPEL7 또는 Fedora 30에 [패키징되지 않은 이유](#)를 다룹니다. [Debian에서도 제거](#)되었습니다.

업스트림 lshe11 프로젝트는 [더 이상 적극적으로 유지 관리되지 않으며 알려진 패치되지 않은](#) 중요 CVE인 [CVE-2016-6902](#) 및 [CVE-2016-6903](#)이 포함되어 있습니다.

Debian 버그에 제안된 대안인 [rssh](#) 역시 유지되지 않는 업스트림이며, 작성자는 수정할 수 없는 보안 문제를 그 이유로 인용합니다.

이러한 이유로 AL2023에 대한 lshe11 추가는 계획되어 있지 않습니다.

monit - 프로세스, 파일, 디렉터리 및 디바이스 모니터

Amazon Linux 2023에서 systemd는 서비스 모니터링, 시작, 중지 및 재시작을 위한 다양한 기능을 제공합니다. 여기에는 속도 제한 재시작, 재시작 시도 간 대기, 장애 시 다른 서비스 시작이 포함됩니다. 자세한 내용은 [systemd.service](#) 설명서를 참조하세요.

AL2023에서 systemd는 경로가 존재하거나 변경되는 경우 같은 이벤트에 대한 작업을 수행하는 데 사용할 수 있는 경로 기반 활성화도 지원합니다. 자세한 내용은 [systemd.path](#) 섹션을 참조하세요.

systemd 단위에는 성공 또는 실패에 대해 수행할 종속성, 조건 및 작업을 지정할 수 있는 일반적인 구성 옵션이 있습니다. 자세한 내용은 [systemd.unit](#) 설명서를 참조하세요.

systemd에서 제공하는 강력한 리소스 제어 기능은 모니터링 태스크에서 과도한 CPU 또는 메모리를 사용하지 않도록 하는 데 사용할 수 있습니다. 자세한 내용은 [systemd.resource-control](#) 섹션을 참조하세요.

nodejs

nodejs 버전 16 패키지는 EPEL7에 있으며 nodejs는 이제 AL2023에 포함됩니다. 작성 당시 AL2023에서 nodejs 버전 18과 20을 모두 사용할 수 있었습니다. 다음 명령을 사용하여 AL2023에 nodejs 18을 설치할 수 있습니다.

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

다음 명령을 사용하여 AL2023에 nodejs 20을 설치할 수 있습니다.

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

perl-Config-General

이제 perl-Config-General 패키지는 EPEL7에 있으며 AL2023에 포함되어 있습니다. 다음 명령을 사용하여 AL2023에 perl-Config-General 패키지를 설치할 수 있습니다.

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

특정 Perl 모듈을 제공하는 패키지를 설치하도록 DNF에 요청하여 Perl 모듈을 설치할 수도 있습니다. 이 방법을 사용하면 OS 패키지 이름 대신 더 친숙한 Perl 모듈 이름을 사용할 수 있습니다.

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

python2-lockfile - 파일 잠금

python2-lockfile 패키지는 EPEL7에 있으며 AL2에는 python-lockfile 패키지가 포함되어 있습니다. AL2023 [Python 2.7은 Python 3으로 대체되었습니다](#).에서는 이 패키지의 Python 2 변형이 AL2023에 추가되지 않습니다.

이 패키지의 Python 3 버전은 AL2023에 포함되어 있습니다. 다음 명령 중 하나를 사용하여 AL2023에 python3-lockfile 패키지를 설치할 수 있습니다.

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

특정 Python 모듈을 제공하는 패키지를 설치하도록 DNF에 요청하여 Python 모듈을 설치할 수도 있습니다.

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

python2-rsa - 순수 Python RSA

python2-rsa 패키지는 EPEL7에 있으며 AL2에는 python2-rsa 패키지가 포함되어 있습니다. AL2023 [Python 2.7은 Python 3으로 대체되었습니다.](#)에서는 이 패키지의 Python 2 변형이 AL2023에 추가되지 않습니다.

이 패키지의 Python 3 버전은 AL2023에 포함되어 있습니다. 다음 명령 중 하나를 사용하여 AL2023에 python3-rsa 패키지를 설치할 수 있습니다.

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

특정 Python 모듈을 제공하는 패키지를 설치하도록 DNF에 요청하여 Python 모듈을 설치할 수도 있습니다.

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

python2-simplejson - Python 2에 대한 JSON 루틴

python2-simplejson 패키지는 EPEL7에 있었습니다. AL2023 [Python 2.7은 Python 3으로 대체되었습니다.](#)에서는 이 패키지의 Python 2 변형이 AL2023에 추가되지 않습니다.

이 패키지의 Python 3 버전은 AL2023에 포함되어 있습니다. 다음 명령을 사용하여 AL2023에 python3-simplejson 패키지를 설치할 수 있습니다.

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

특정 Python 모듈을 제공하는 패키지를 설치하도록 DNF에 요청하여 Python 모듈을 설치할 수도 있습니다.

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

rkhunter - Rootkit Hunter

rkhunter 패키지는 chkrootkit와 함께 AL2023에 포함되어 있습니다.

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

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

rsync - OpenSSH와 함께 사용할 수 있는 제한된 셸

rsync 패키지는 EPEL7에 있었습니다. 업스트림 [rsync](#) 패키지는 유지 관리되지 않으며 작성자는 수정할 수 없는 보안 문제를 그 이유로 인용합니다.

작성자가 수정할 수 없는 보안 문제를 인용하고 있으며, rsync를 AL2023에 추가하는 것은 계획되어 있지 않습니다.

sscg - 자체 서명된 SSL 인증서 생성기

sscg 패키지는 AL2 Core와 EPEL7에 있었습니다. sscg 패키지는 AL2023의 일부이기도 합니다.

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

stress - 스트레스 테스트

stress 패키지는 EPEL7에 있으며 AL1에서도 사용할 수 있습니다.

stress 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

stress-ng - 스트레스 테스트

stress-ng 패키지는 EPEL7에 있으며 testing AL2 Extra에서도 사용할 수 있습니다.

stress-ng 패키지는 AL2023에 포함되어 있으며 다음 명령을 실행하여 설치할 수 있습니다.

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

tmpwatch - 마지막으로 액세스한 시간을 기준으로 파일 제거

Amazon Linux 2023에서 이 기능은 [systemd-tmpfiles](#)에서 제공됩니다.

xmlstarlet - 명령줄 XML 유틸리티

xmlstarlet 패키지는 EPEL7에 있으며 AL2023에서는 사용할 수 없습니다.

업스트림 패키지는 9년 동안 터치되지 않았습니다(2014년 8월에 마지막으로 터치됨). 추가로 4년 전(최소 2010년 7월 이후)에 새 유지 관리에 대한 요청에 응답하지 않았습니다. 따라서 AL2023에 xmlstarlet를 추가할 계획이 없습니다.

Python 2.7은 Python 3으로 대체되었습니다.

AL2 코어 패키지에 대한 장기 지원(LTS) 약속의 일환으로 2025년 6월까지 Python 2.7 지원 및 보안 패치를 제공합니다. 업스트림 Python 커뮤니티가 공지한 2020년 1월 Python 2.7 서비스 종료 이후에도 지원합니다.

AL2는 Python 2.7에 상당히 종속되는 yum 패키지 관리자를 사용합니다. AL2023 버전의 dnf 패키지 관리자가 Python 3으로 마이그레이션되어 더 이상 Python 2.7이 필요하지 않습니다. AL2023은 Python 3으로 완전히 전환되었습니다.

Note

AL2023에서 Python 2.7이 제거되었으므로, Python을 사용하는 OS 구성 요소는 Python 3에서 작동됩니다. Amazon Linux가 제공하고 지원하는 Python 버전을 계속 사용하려면 Python 2 코드를 Python 3로 변환하세요.

Amazon Linux Python에 대한 자세한 내용은 [AL2023에서 Python 사용](#)을 참조하세요.

보안 업데이트

Amazon Linux 2023은 AL2에 있는 하드닝을 개선합니다. 자세한 내용은 [Amazon Linux 2023 보안 및 규정 준수](#) 단원을 참조하십시오. AL2의 커널 하드닝 변경 사항에 대한 자세한 내용은 [보안 중심으로 커널 구성 변경](#) 섹션을 참조하세요.

주제

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

SELinux

기본적으로 AL2023 Security Enhanced Linux (SELinux)는 enabled되어 있으며 permissive 모드로 설정되어 있습니다. permissive 모드에서 권한 거부는 기록되지만 강제하지는 않습니다.

SELinux는 AL2 disabled에 있던 Amazon Linux 커널의 보안 기능입니다. SELinux는 커널의 주요 하위 시스템에 필수 액세스 제어(MAC) 아키텍처를 제공하는 커널 기능 및 유틸리티입니다.

자세한 내용은 [AL2023의 SELinux 모드 설정](#) 단원을 참조하십시오.

SELinux 리포지토리, 도구 및 정책에 대한 자세한 내용은 [SELinux 노트북](#), [SELinux 정책 유형](#) 및 [SELinux 프로젝트](#)를 참조하세요.

OpenSSL 3

AL2023에는 Open Secure Sockets Layer version 3 (OpenSSL 3) 암호화 툴킷이 있습니다. AL2023은 TLS 1.3 및 TLS 1.2 네트워크 프로토콜을 지원합니다.

기본적으로 AL2는 OpenSSL 1.0.2과 함께 제공됩니다. OpenSSL 1.1.1에 애플리케이션을 빌드할 수 있습니다.

OpenSSL에 대한 자세한 내용은 [OpenSSL 마이그레이션 가이드](#)를 참조하세요.

보안에 대한 자세한 내용은 [보안 업데이트 및 기능](#)을 참조하세요.

IMDSv2

기본적으로 AL2023 AMI에서 실행하는 인스턴스는 IMDSv2만 요구하며 홉 제한이 2로 설정되어서 컨테이너 워크로드를 지원합니다. `imds-support` 파라미터를 `v2.0`로 설정하면 됩니다. 자세한 내용은 Amazon EC2 사용 설명서의 [AMI 구성](#) 섹션을 참조하세요.

Note

세션 토큰의 유효 시간은 1초에서 6시간 사이입니다. IMDSv2 쿼리에 API 요청을 전달하는 주소는 다음과 같습니다.

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

이 설정을 수동으로 재정의하여 인스턴스 메타데이터 옵션 시작 속성을 사용하여 IMDSv1을 활성화할 수 있습니다. 또한 IAM을 컨트롤하여 다른 IMDS 설정을 적용할 수 있습니다. 인스턴스 메타데이터 서비스 설정 및 사용에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 [IMDSv2 사용](#), [새 인스턴스에 대한 인스턴스 메타데이터 옵션 구성](#), [기존 인스턴스의 인스턴스 메타데이터 옵션 수정](#) 섹션을 참조하세요.

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

Note

AL2023에는 log4j-cve-2021-44228-hotpatch 패키지가 없습니다.

[CVE-2021-44228](#)을 위해 AL1 및 AL2용 [Apache Log4j용 핫패치](#) RPM 패키지 버전을 출시했습니다. [Amazon Linux에 핫패치를 추가한다고 발표](#)하면서 “핫패치 설치가 CVE-2021-44228 또는 CVE-2021-45046 완화 기능을 갖춘 log4j 버전 업데이트를 대체하는 것이 아니라고 언급했습니다.

이 핫패치는 log4j 패치 시간을 벌기 위한 조치였습니다. [CVE-2021-44228](#)에 대응한 지 15개월 후에 첫 AL2023 정식 릴리스(GA)가 이루어졌기 때문에 AL2023에 핫패치 (활성화 여부와 관계없이)를 포함되어 있지 않습니다.

사용자는 Amazon Linux에서 자체 log4j 버전을 실행한다면 [CVE-2021-44228](#) 또는 [CVE-2021-45046](#)의 영향을 받지 않는 버전으로 업데이트해야 합니다.

AL2023은 보안 패치를 최신 상태로 유지할 수 있도록 [AL2023 업데이트](#) 지침을 제공합니다. 보안 권고는 [Amazon Linux 보안 센터](#)에 나와 있습니다.

안정성을 위한 결정적 업그레이드

버전이 지정된 리포지토리 기능으로 결정적 업그레이드 기능을 사용하면 모든 AL2023 AMI는 특정 리포지토리 버전에서만 사용할 수 있게 됩니다. 결정적 업그레이드를 통해 패키지 버전과 업데이트 간의 일관성을 높일 수 있습니다. 메이저 또는 마이너 릴리스에는 특정 리포지토리 버전이 포함되어 있습니다.

최신 AL2023 버전에는 결정적 업그레이드가 기본적으로 활성화되어 있습니다. 이는 AL2 및 기타 구 버전에서 사용되었던 수동 종본 잠금 방법에서 개선된 방식입니다.

자세한 내용은 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#) 단원을 참조하십시오.

기본 Amazon EBS 볼륨 유형인 gp3

AL2023 AMI와 AL2 모두 루트 파일 시스템의 XFS 파일 시스템을 사용합니다. 루트 디바이스 파일 시스템 mkfs 옵션은 AL2023 Amazon EC2에 맞게 최적화되었습니다. 또한 AL2023은 특정 요구 사항에 따라 다른 볼륨에서 사용할 수 있는 여러 다른 파일 시스템을 지원합니다.

AL2023 AMI는 기본적으로 Amazon EBS gp3 볼륨을 사용하는 반면, AL2 AMI는 기본적으로 Amazon EBS gp2 볼륨을 사용합니다. 인스턴스 시작 시 볼륨을 변경할 수 있습니다.

Amazon EBS 볼륨 유형에 대한 자세한 내용은 [Amazon EBS 범용 볼륨](#)(Amazon EBS General Purpose Volumes)을 참조하세요.

Amazon EC2 인스턴스 시작에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [인스턴스 시작](#) 섹션을 참조하세요.

통합 제어 그룹 계층 구조 (cgroup v2)

제어 그룹(cgroup)은 프로세스를 계층적으로 구성하고 프로세스 간에 시스템 리소스를 배포하는 Linux 커널 기능입니다. 제어 그룹은 systemd로 컨테이너 런타임을 광범위하게 구현할 수 있습니다.

AL2는 cgroupv1을 지원하고 AL2023은 cgroupv2를 지원합니다. 이는 컨테이너 워크로드를 실행할 때 특히 유용합니다(예: [AL2023 기반 Amazon ECS AMI를 사용하여 컨테이너화된 워크로드 호스팅](#)).

AL2023에서 cgroupv1로 시스템 실행할 수 있는 코드를 계속 사용할 수 있지만 이 구성은 권장되거나 지원되지 않으며 향후 Amazon Linux 주요 릴리스에서 완전히 제거될 예정입니다.

[저사양 Linux 커널 인터페이스](#)에 관한 광범위한 설명서와 [systemd cgroup 위임 설명서](#)가 있습니다.

컨테이너 이외의 일반 사용 사례는 사용할 수 있는 시스템 리소스에 제한이 있는 systemd 유닛을 만드는 것입니다. 자세한 내용은 [systemd.resource-control](#) 섹션을 참조하세요.

systemd 타이머 교체 cron

cronie 패키지는 AL2 AMI에 기본적으로 설치되었으며, 정기 작업을 예약하는 기존 crontab 방식을 지원합니다. AL2023에서는 cronie이 기본적으로 포함되지 않습니다. 따라서 crontab에 대한 지원은 더 이상 기본적으로 제공되지 않습니다.

cronie 패키지를 옵션으로 설치하여 기존 cron 작업을 할 수 있습니다. systemd에서 추가 기능을 제공하므로 systemd 타이머로 마이그레이션하는 것이 좋습니다.

향상된 도구 체인: gcc, binutils 및 glibc

AL2023에는 AL2와 동일한 코어 패키지가 다수 포함되어 있습니다.

AL2023에 다음 세 가지 핵심 툴체인 패키지를 업데이트했습니다.

패키지 이름	AL2	AL2023
glibc	2.26	2.34
gcc	7.3	11.3
binutils	2.29	2.39

자세한 내용은 [코어 툴체인 패키지 glibc, gcc 및 binutils](#) 단원을 참조하십시오.

업데이트된 기본 언어 표준을 포함하여 C, C++ 및 Fortran 언어 런타임에 대한 자세한 내용은 [AL2023의 C, C++, 및 Fortran](#) 섹션을 참조하세요.

최적화에 대한 자세한 내용은 [성능 및 운영 최적화](#)를 참조하세요.

systemd 저널 대체 rsyslog

AL2023 로깅 시스템 패키지는 AL2에서 업그레이드되었습니다. AL2023에 rsyslog가 설치되어 있지 않으므로 AL2에 있던 `/var/log/messages`와 같은 텍스트 기반 로그 파일은 사용할 수 없습니다. AL2023 기본 구성은 systemd-journal이며 journalctl으로 검사할 수 있습니다. rsyslog는 AL2023 옵션 패키지이지만 새로운 systemd 기반 journalctl 인터페이스와 관련 패키지를 사용하는 것이 좋습니다. 자세한 내용은 [journalctl](#) 매뉴얼 페이지를 참조하세요.

일반적으로 사용되는 몇 가지 syslog 명령에 해당하는 systemd journal은 다음 표에서 다룹니다.

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

패키지 종속성 최소화

Amazon Linux 2023은 많은 패키지의 종속성 그래프를 최소화하여 애플리케이션에 더 작은 공간을 제공합니다. AL2의 주목할 만한 변경 사항에는 일반적으로 사용되는 기능을 유지하면서 필요한 패키지 수를 크게 줄이는 `curl-minimal` 및 `gnupg-minimal` 패키지가 포함됩니다.

주제

- [curl 및 libcurl 패키지 변경](#)
- [GNU 프라이버시 가드\(GNUPG\)](#)

curl 및 libcurl 패키지 변경

AL2023에서 `curl` 및 `libcurl` 패키지의 공통 프로토콜과 기능을 `curl-minimal` 및 `libcurl-minimal`로 구분할 수 있습니다. 이는 AL2023 AMI 및 컨테이너 기본 패키지이며 대부분 사용자의 디스크, 메모리 및 종속성 풋프린트 용량을 줄입니다.

`gopher://`를 지원하기 위해 `curl`의 전체 기능이 필요한 경우 다음 명령을 실행하여 `curl-full` 및 `libcurl-full` 패키지를 설치하세요.

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

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

GNU 프라이버시 가드(GNUPG)

AL2023은 `gnupg2` 패키지의 최소 기능과 전체 기능을 `gnupg2-minimal`와 `gnupg2-full` 패키지로 구분합니다. `gnupg2-minimal` 패키지가 기본적으로 설치되어 있습니다. 이는 rpm 패키지의 디지털 서명을 확인하는 데 필요한 최소한의 기능을 제공합니다.

키 서버에서 키를 다운로드하기 등 더 많은 기능을 사용하려면 `gnupg2-full` 패키지가 설치되어 있어야 합니다. `gnupg2` `gnupg2-minimal`을 `gnupg2-full`로 변경하려면 다음 명령을 실행합니다.

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

기본 JVM으로 Amazon Corretto 설정

AL2023에는 기본 (그리고 유일한) Java 개발 키트(JDK)로 [Amazon Corretto](#)를 사용할 수 있습니다. AL2023에 설치된 모든 Java 기반 패키지는 모두 Amazon Corretto 17로 빌드됩니다.

AL2에서 마이그레이션하는 경우 AL2의 해당 OpenJDK 버전에서 Amazon Corretto로 원활하게 전환할 수 있습니다.

AWS CLIV2

AL2023은 AWS CLI버전 2와 함께 제공되는 반면, AL2는 버전 1과 함께 제공됩니다AWS CLI.

UEFI 기본 설정 및 보안 부팅

기본적으로 UEFI 펌웨어를 지원하는 인스턴스 유형에서 AL2023 AMI로 실행된 인스턴스는 UEFI 모드에서 시작하려면 부팅 모드 AMI 파라미터를 `uefi-preferred`로 설정하면 됩니다. 자세한 설명은 Amazon EC2 사용 설명서의 [부팅 모드](#) 섹션을 참조하세요.

UEFI 보안 부팅을 지원하는 Amazon EC2 인스턴스 유형에서는 Amazon Linux 2023에서 보안 부팅을 활성화할 수 있습니다. 자세한 내용은 [AL2023의 UEFI 보안 부팅](#) 단원을 참조하십시오.

SSH 서버 기본 구성 변경

AL2023 AMI에서 릴리스에서 생성하는 sshd 호스트 키 유형을 변경했습니다. 또한 실행 시 키가 생성되지 않도록 일부 레거시 키 유형을 삭제했습니다. 클라이언트는 `rsa-sha2-256` 및 `rsa-sha2-512` 프로토콜 또는 `ed25519` 키를 사용하는 `ssh-ed25519`를 지원해야 합니다. 기본적으로 `ssh-rsa` 서명은 비활성화되어 있습니다.

또한 기본 `sshd_config` 파일의 AL2023 구성 설정에는 `UseDNS=no`가 포함되어 있습니다. 이렇게 설정하면 DNS 장애로 인해 인스턴스와 ssh 세션을 설정하는 기능이 차단될 가능성이 줄어듭니다. 단점은 `authorized_keys` 파일의 `from=hostname.domain,hostname.domain` 줄 항목이 확인되지 않는다는 것입니다. sshd은 더 이상 DNS 이름을 확인하지 않으므로 심표로 구분된 각 `hostname.domain` 값을 해당 IP address 값으로 변환해야 합니다.

자세한 내용은 [SSH 서버 기본 구성](#) 단원을 참조하십시오.

AL2 이후 AL2023 커널 변경 사항

AL2023은 6.1 커널과 함께 Amazon Linux를 클라우드용으로 최적화하기 위해 여러 구성을 변경했습니다. 대부분의 사용자의 경우 이러한 변경 사항은 완전히 투명해야 합니다.

IPv4 TTL

IPv4용 TTL은 `sysctl`을 통해 구성되며 기본값은 `/etc/sysctl.d/00-defaults.conf`에 있습니다. 이 값은 일반적인 `sysctl` 방법을 통해 사용자 지정할 수 있습니다. 자세한 내용은 `sysctl man` 페이지를 참조하세요.

AL2는 `net.ipv4.ip_default_ttl` 값을 255로 설정하고 AL2023은 값을 127로 설정합니다. 이렇게 하면 Amazon Linux 기본값이 다른 주요 Linux 배포판과 일치합니다. 증명된 필요 없이 이 기본값을 변경하는 것은 권장되지 않습니다.

보안 중심으로 커널 구성 변경

CONFIG 옵션	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_GONDOLOON_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 옵션	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_RTIFY_RCE	n	y	n	y	y	y	y	y	y	y
CONFIG_RDENED_ERCOPY_LLBACK	해당 사항 없음	해당 사항 없음	y	y	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_IT_ON_OC_DEFAULT_ON	해당 사항 없음	해당 사항 없음	n	n	n	n	n	n	n	n
CONFIG_IT_ON_E_DEFAULT_ON	해당 사항 없음	해당 사항 없음	n	n	n	n	n	n	n	n
CONFIG_MMU_DELT_DMA_RICT	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	n	n	n	n	n	n
CONFIG_ISC_ALOAD	y	y	y	y	n	n	n	n	n	n
CONFIG_HED_CO	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y

CONFIG 옵션	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_HED_START_END_CHECK	n	y	n	y	y	y	y	y	y	y
CONFIG_CURITY_ESG_REACT	n	n	n	n	y	y	y	y	y	y
CONFIG_CURITY_LINUX_ABLE	y	y	y	y	n	n	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_UFFLE_E_ALLOC_OR	해당 사항 없음	해당 사항 없음	y	y	y	y	y	y	y	y
CONFIG_AB_FREEST_HARDED	n	y	y	y	y	y	y	y	y	y
CONFIG_AB_FREEST_RAN	n	n	y	y	y	y	y	y	y	y

x86-64 특정 보안 중심 커널 구성 변경

CONFIG 옵션	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	해당 사항 없음	해당 사항 없음
CONFIG_RANDOMIZE_MEMORY	해당 사항 없음	y	y	y	y

aarch64(ARM/Graviton) 특정 보안 중심 커널 구성 변경

CONFIG 옵션	AL2/4.14/ aarch64	AL2/5.10/ aarch64	AL2023/6.1/ aarch64	AL2023/6.12/ aarch64	AL2023/6.18/ aarch64
CONFIG_ARM64_PTR_AUTH	해당 사항 없음	y	y	y	y
CONFIG_ARM64_PTR_AUTH_KERNEL	해당 사항 없음	해당 사항 없음	y	y	y
CONFIG_ARM64_SW_TTBR0_PAN	y	y	y	y	y

/dev/mem, /dev/kmem 및 /dev/port

Amazon Linux 2023은 AL2에 이미 적용되어 있는 제한을 기반으로 하여 /dev/mem와 /dev/port(CONFIG_DEVMEM 및 CONFIG_DEVPORT)를 완전히 비활성화합니다.

이 /dev/kmem 코드는 Linux 커널 5.13에서 완전히 제거되었지만 AL2에서는 비활성화되었으며 이제 AL2023에는 전혀 사용할 수 없습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

FORTIFY_SOURCE

지원되는 모든 아키텍처에서는 AL2023에 의해 CONFIG_FORTIFY_SOURCE를 사용할 수 있습니다. 이 기능은 보안 하드닝 기능입니다. 컴파일러가 버퍼 크기를 확인하고 검증할 수 있는 경우 이 기능은 공통 문자열 및 메모리 함수의 버퍼 오버플로를 탐지할 수 있습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

라인 디시플린 자동 로드(CONFIG_LDISC_AUTOLOAD)

AL2023 커널은 CAP_SYS_MODULE 권한을 가진 프로세스의 요청이 아니면, 소프트웨어가 TIOCSETD ioctl을 사용하는 경우처럼 라인 디시플린을 자동으로 로드하지 않습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

권한이 없는 사용자를 위한 dmesg 액세스(CONFIG_SECURITY_DMESG_RESTRICT)

기본적으로 AL2023은 권한이 없는 사용자에게 dmesg에 대한 액세스를 허용하지 않습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

SELinux selinuxfs 비활성화

AL2023은 더 이상 사용되지 않는 CONFIG_SECURITY_SELINUX_DISABLE 커널 옵션을 비활성화하여 정책을 로드하기 전에 SELinux를 비활성화하는 런타임 메서드를 활성화했습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

기타 커널 구성 변경

CONFIG 옵션	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										
CONFIG	4096	8192	4096	8192	4096	8192	4096	8192	4096	8192
CONFIG	y	n	y	n	y	y	y	y	y	y
CONFIG										
CONFIG	1	0	1	0	1	1	1	1	해당 사항 없음	해당 사항 없음
CONFIG	m	m	m	m	m	m	m	m	m	m
CONFIG	m	m	m	m	n	n	n	n	n	n
CONFIG	해당 사항 없음	y	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n

CONFIG_HZ

AL2023은 x86-64 및 aarch64 플랫폼 모두에서 CONFIG_HZ를 100으로 설정합니다.

CONFIG_NR_CPUS

AL2023에서 CONFIG_NR_CPUS를 Amazon EC2에 있는 최대 CPU 코어 수와 비슷한 숫자로 설정합니다.

Panic on OOPS

AL2023 커널이 실패하면 패닉 상태가 됩니다. 이 기능은 커널 명령줄에서 oops=panic로 부팅하는 것과 같습니다.

커널 오프스는 커널이 시스템 신뢰성에 영향을 미칠 수 있는 내부 오류를 감지합니다.

PPP 및 SLIP 지원

AL2023은 SLIP 프로토콜을 지원하지 않지만 최신 AL2023 커널에서 PPP 프로토콜을 지원할 수 있습니다.

Xen PV 게스트 지원

AL2023은 Xen PV 게스트 실행을 지원하지 않습니다.

커널 파일 시스템 지원

AL2 커널에서 마운트를 지원하는 파일 시스템이 몇 가지 변경되었으며 구문 분석할 파티션 스키마도 변경되었습니다.

CONFIG 옵션	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	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 옵션	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_AMFS	m	m	m	m	n	n	n	n	n	n
CONFIG_AMFS_BLOCKDEV	해당 사항 없음	해당 사항 없음	y	n	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_CLONE	해당 사항 없음	해당 사항 없음	n	n	n	n	n	n	n	n
CONFIG_ERA	m	n	m	n	n	n	n	n	n	n
CONFIG_INTEGRITY	n	m	n	m	m	m	m	m	m	m
CONFIG_LOG_WRITES	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_CRYPTO_FIPS	n	m	n	m	n	n	n	n	n	n
CONFIG_FAT_FS	해당 사항 없음	해당 사항 없음	m	m	m	m	m	m	m	m

CONFIG 옵션	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_T2_FS	n	m	n	m	n	n	n	n	n	n
CONFIG_T3_FS	n	m	n	m	n	n	n	n	해당 사항 없음	해당 사항 없음
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_S_FS	n	n	n	n	n	n	n	n	n	n
CONFIG_M_PARTITION	n	y	n	y	n	n	n	n	n	n
CONFIG_C_PARTITION	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 옵션	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_MFS_FS	n	m	n	m	n	n	n	n	n	n
CONFIG_LARIS_PARTITION	n	y	n	y	n	n	n	n	n	n
CONFIG_UASHFSTD	n	y	n	y	y	y	y	y	y	y
CONFIG_N_PARTITION	n	y	n	y	n	n	n	n	n	n

앤드류 파일 시스템 지원(AFS)

커널은 더 이상 afs 파일 시스템을 지원하도록 빌드되지 않습니다. AL2에서는 afs 사용자 공간을 지원하지 않습니다.

cramfs 지원

커널은 더 이상 cramfs 파일 시스템을 지원하도록 빌드되지 않습니다. squashfs 파일 시스템은 AL2023의 후속 버전입니다.

BSD 디스크 레이블 지원

커널은 더 이상 BSD 디스크 레이블을 지원하도록 빌드되지 않습니다. BSD 디스크 레이블이 있는 볼륨을 읽어야 하는 경우 여러 BSD를 실행할 수 있습니다.

Device Mapper 변경 사항

AL2023 커널에 구성된 디바이스 매퍼 대상에 대해 몇 가지 변경 사항이 있습니다.

eCryptFs 지원

ecryptfs 파일 시스템은 Amazon Linux에서 더 이상 사용할 수 없습니다. `ecryptfs`의 사용자 공간 구성 요소는 AL1에 존재했었고 AL2에서 삭제되었으며 AL2023 커널은 `ecryptfs`를 지원하는 커널을 빌드하지 않습니다.

exFAT

exFAT 파일 시스템에 대한 지원이 AL2의 5.10 커널에 추가되었습니다. 이것은 4.14 커널을 사용한 AL2 시작 시 존재하지 않았습니다. AL2023은 exFAT 파일 시스템을 계속 지원합니다.

ext2, ext3 및 ext4 파일 시스템

AL2023에 `CONFIG_EXT4_USE_FOR_EXT2` 옵션이 있습니다. 즉, `ext4` 파일 시스템 코드를 사용하여 레거시 `ext2` 파일 시스템을 읽을 수 있습니다.

CONFIG_GFS2_FS

커널은 더 이상 `CONFIG_GFS2_FS`로 빌드되지 않습니다.

Apple Extended HFS 파일 시스템 지원 (HFS+)

AL2에서는 `hfsplus` 파일 시스템 지원을 통해 x86-64 커널만 빌드되었습니다. AL2 5.15 커널에는 아키텍처에 대한 `hfsplus` 지원이 포함되지 않습니다. AL2023 Amazon Linux의 `hfsplus` 지원이 중단되었습니다.

HFS 파일 시스템 지원

AL2에서는 `hfs` 파일 시스템 지원을 통해 x86-64 커널만 빌드되었습니다. AL2 5.15 커널에는 아키텍처에 대한 `hfs` 지원이 포함되지 않습니다. AL2023 Amazon Linux의 `hfs` 지원이 중단되었습니다.

JFS 파일 시스템 지원

이전 x86-64 AL2 커널은 `jfs` 파일 시스템 지원을 통해 빌드되었습니다. AL2 5.15 커널에는 아키텍처에 대한 `jfs` 지원이 포함되지 않습니다. AL1과 AL2에 JFS 사용자 공간이 제공되지 않았습니다. AL2023 Amazon Linux의 `jfs` 지원이 중단되었습니다.

업스트림 Linux 커널은 [JFS 제거를 고려하고 있습니다](#). 따라서 JFS 파일 시스템에 데이터가 있는 경우 다른 파일 시스템으로 마이그레이션해야 합니다. 2024년에 JFS는 현재 모든 Amazon Linux 커널에서 제거되었습니다.

Windows 논리 디스크 관리자(동적 디스크) 지원(**CONFIG_LDM_PARTITION**)

AL2023은 MS-DOS 스타일 파티션이 있는 Windows 2000, Windows XP 또는 Windows Vista 동적 디스크를 더 이상 지원하지 않습니다. 이 코드는 Windows Vista에 도입된 최신 GPT 기반 동적 디스크를 지원하지 않았습니다.

Macintosh 파티션 맵 지원

AL2023은 더 이상 기존 Macintosh 파티션 맵을 지원하지 않습니다. 최신 macOS 버전은 구 버전의 최신 GPT 파티션 테이블을 생성합니다.

NFSv2 지원

AL2023은 더 이상 NFSv2를 지원하지 않지만 NFSv3, NFSv4, NFSv4.1 및 NFSv4.2는 계속 지원합니다. NFSv3 이후 버전으로 마이그레이션하실 것을 권장합니다.

NTFS(**CONFIG_NTFS_FS**)

AL2 커널 5.10부터 Amazon Linux의 NTFS 파일 시스템에 액세스하는 `ntfs3` 코드가 `ntfs`로 변경되었습니다. AL2023에는 더 이상 `ntfs` 코드가 포함되지 않으며 NTFS 파일 시스템에 액세스하기 위한 `ntfs3` 코드에만 있습니다.

romfs 파일 시스템

`squashfs` 파일 시스템은 Amazon Linux `romfs` 파일 시스템의 후속 버전이며, AL2023 커널은 더 이상 `romfs`를 지원하지 않습니다.

Solaris x86 하드 디스크 파티션 형식

AL2023은 더 이상 Solaris x86 하드 디스크 파티션 형식을 지원하지 않습니다.

squashfs zstd 압축

AL2023에 모든 지원 아키텍처에서 zstd 압축 `squashfs` 파일 시스템에 대한 지원이 추가되었습니다.

Sun 파티션 테이블 지원

AL2023에 더 이상 Sun 파티션 테이블 형식(**CONFIG_SUN_PARTITION**)에 대한 지원이 포함되지 않습니다.

/tmp가 이제 tmpfs입니다.

Amazon Linux 2023은 Amazon Linux 2와 비교할 때 /tmp의 동작 방식에 대한 변경 사항을 도입합니다. AL2의 기본 구성은 /tmp 및 /var/tmp가 모두 루트 파일 시스템에 있다는 것입니다. Amazon Linux 2023은 기본적으로 RAM의 50% 및 최대 100만 inodes로 /tmp에 tmpfs를 사용합니다. 이러한 변경 사항은 Amazon Linux를 다른 Linux 배포판의 동작과 일치시킵니다.

AL2023의 파일 시스템 레이아웃에 대한 자세한 내용은 [파일 시스템 레이아웃](#) 섹션의 [/tmp](#) 및 [/var/tmp](#) 부분을 참조하세요.

AMI 및 컨테이너 이미지 변경

AMI 및 컨테이너에 포함된 패키지에 몇 가지 변경 사항이 있습니다.

Amazon Linux 2023에는 [the section called “기본 AL2023 컨테이너 이미지 빌드”](#) 빌드에 대한 [the section called “AL2023 최소 컨테이너 이미지”](#) 및 지원이 도입되었습니다. 자세한 내용은 [컨테이너에서 AL2023 사용](#) 단원을 참조하십시오.

Amazon Linux 2와 Amazon Linux 2023 AMI에 설치된 패키지 비교

Amazon Linux 2 및 AL2023 표준 AMI에 있는 RPM 간의 비교입니다.

패키지	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

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
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
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	

패키지	AL2 AMI	AL2023 AMI
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	
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

패키지	AL2 AMI	AL2023 AMI
dracut-config-ec2	2.0	3.0
dracut-config-generic	033	055
dwz		0.14
dyninst	9.3.1 (x86_64)	10.2.1
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

패키지	AL2 AMI	AL2023 AMI
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
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	

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

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
hunspell-filesystem		1.7.0
hwdata	0.252	0.384
info	5.1	6.7
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	2.10	2.14
jbigkit-libs	2.0	
jemalloc		5.2.1
jitterentropy		3.4.1
jq		1.7.1
json-c	0.11	0.14
kbd	1.15.5	2.4.0
kbd-legacy	1.15.5	
kbd-misc	1.15.5	2.4.0
kernel	5.10.228	6.1.112

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
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
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	

패키지	AL2 AMI	AL2023 AMI
libdwarf	20130207 (x86_64)	
libeconf		0.4.0
libedit	3.0	3.1
libestr	0.1.9	
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	

패키지	AL2 AMI	AL2023 AMI
libkcapi		1.4.0
libkcapi-hmaccalc		1.4.0
libldb		2.6.2
libmaxminddb		1.5.2
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
libnghttp2	1.41.0	1.59.0
libnl3	3.2.28	3.5.0
libnl3-cli	3.2.28	
libpath_utils	0.2.1	0.2.1
libpcap	1.5.3	1.10.1
libpciaccess	0.14 (x86_64)	
libpipeline	1.2.3	1.5.3
libpkgconf		1.8.0
libpng	1.5.13	

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
libstoragegmt	1.6.1	1.9.4
libstoragegmt-python	1.6.1	
libstoragegmt-python-clibs	1.6.1	
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

패키지	AL2 AMI	AL2023 AMI
libverto-libevent	0.2.5	
libwebp	0.3.0	
libxcrypt		4.4.33
libxml2	2.9.1	2.10.4
libxml2-python	2.9.1	
libyaml	0.1.4	0.2.5
libzstd		1.5.5
linux-firmware-whence		20210208(noarch)
lm_sensors-libs	3.4.0	3.6.0
lmdb-libs		0.9.29
logrotate	3.8.6	3.20.1
lsof	4.87	4.94.0
lua	5.1.4	
lua-libs		5.4.4
lua-srpm-macros		1
lvm2	2.02.187	
lvm2-libs	2.02.187	
lz4	1.7.5	
lz4-libs		1.9.4
make	3.82	

패키지	AL2 AMI	AL2023 AMI
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	
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

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
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
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

패키지	AL2 AMI	AL2023 AMI
perl-File-Basename		2.85
perl-File-Path	2.09	2.18
perl-File-stat		1.09
perl-File-Temp	0.23.01	0.231.100
perl-Filter	1.49	
perl-Getopt-Long	2.40	2.52
perl-Getopt-Std		1.12
perl-HTTP-Tiny	0.033	0.078
perl-if		0.60.800
perl-interpreter		5.32.1
perl-IIO		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

패키지	AL2 AMI	AL2023 AMI
perl-Pod-Escapes	1.04	1.07
perl-podlators	2.5.1	4.14
perl-Pod-Perldoc	3.20	3.28.01
perl-Pod-Simple	3.28	3.42
perl-Pod-Usage	1.63	2.01
perl-POSIX		1.94
perl-Scalar-List-Utils	1.27	1.56
perl-SelectSaver		1.02
perl-Socket	2.010	2.032
perl-srpm-macros		1
perl-Storable	2.45	3.21
perl-subs		1.03
perl-Symbol		1.08
perl-Term-ANSIColor		5.01
perl-Term-Cap		1.17
perl-Text-ParseWords	3.29	3.30
perl-Text-Tabs+Wrap		2021.0726
perl-threads	1.87	
perl-threads-shared	1.43	
perl-Time-HiRes	1.9725	

패키지	AL2 AMI	AL2023 AMI
perl-Time-Local	1.2300	1.300
perl-vars		1.05
pinentry	0.8.1	
pkgconf		1.8.0
pkgconfig	0.27.1	
pkgconf-m4		1.8.0
pkgconf-pkg-config		1.8.0
plymouth	0.8.9	
plymouth-core-libs	0.8.9	
plymouth-scripts	0.8.9	
pm-utils	1.4.1	
polycoreutils	2.5	3.4
polycoreutils-python-utils		3.4
popt	1.13	1.18
postfix	2.10.1	
procps-ng	3.3.10	3.3.17
protobuf-c		1.4.1
psacct	6.6.1	6.6.4
psmisc	22.20	23.4
pth	2.0.7	

패키지	AL2 AMI	AL2023 AMI
publicsuffix-list-dafsa	20240208	20240212
pygpgme	0.3	
pyliblzma	0.5.3	
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

패키지	AL2 AMI	AL2023 AMI
python3-attrs		20.3.0
python3-audit		3.0.6
python3-awscrt		0.19.19
python3-babel		2.9.1
python3-cffi		1.14.5
python3-chardet		4.0.0
python3-colorama		0.4.4
python3-configobj		5.0.6
python3-cryptography		36.0.1
python3-daemon	2.2.3	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

패키지	AL2 AMI	AL2023 AMI
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	3.7.16	3.9.16
python3-libselenium		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

패키지	AL2 AMI	AL2023 AMI
python3-prompt-toolkit		3.0.24
python3-pycparser		2.20
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-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

패키지	AL2 AMI	AL2023 AMI
python3-urllib3		1.25.10
python3-wcwidth		0.2.5
python-babel	0.9.6	
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	

패키지	AL2 AMI	AL2023 AMI
python-kitchen	1.1.1	
python-libs	2.7.18	
python-lockfile	0.9.1	
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	

패키지	AL2 AMI	AL2023 AMI
readline	6.2	8.1
rng-tools	6.8	6.14
rootfiles	8.1	8.1
rpcbind	0.2.0	1.2.6
rpm	4.11.3	4.16.1.3
rpm-build-libs	4.11.3	4.16.1.3
rpm-libs	4.11.3	4.16.1.3
rpm-plugin-selinux		4.16.1.3
rpm-plugin-systemd-inhibit	4.11.3	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

패키지	AL2 AMI	AL2023 AMI
setserial	2.17	
setup	2.8.71	2.13.7
setuptools	1.19.11	
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

패키지	AL2 AMI	AL2023 AMI
systemd-pam		252.23
systemd-resolved		252.23
systemd-sysv	219	
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	

패키지	AL2 AMI	AL2023 AMI
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.30.2	2.37.4
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	

패키지	AL2 AMI	AL2023 AMI
yum	3.4.3	4.14.0
yum-langpacks	0.4.2	
yum-metadata-parser	1.1.4	
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 및 Amazon Linux 2023 미니멀 AMI에 설치된 패키지 비교

Amazon Linux 2 및 AL2023 최소 AMI에 있는 RPM 간의 비교입니다.

패키지	AL2 최소	AL2023 최소
acl	2.2.51	
alternatives		1.15
amazon-chrony-config		4.3
amazon-ec2-net-utils		2.5.1

패키지	AL2 최소	AL2023 최소
amazon-linux-extras	2.0.3	
amazon-linux-repo-s3		2023.6.20241031
amazon-linux-sb-keys		2023.1
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

패키지	AL2 최소	AL2023 최소
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	
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	

패키지	AL2 최소	AL2023 최소
dhcp-libs	4.2.5	
diffutils	3.3	3.8
dnf		4.14.0
dnf-data		4.14.0
dnf-plugin-release-notification		1.2
dnf-plugins-core		4.3.0
dnf-plugin-support-info		1.2
dracut	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

패키지	AL2 최소	AL2023 최소
elfutils-libelf	0.176	0.188
elfutils-libs	0.176	0.188
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	
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

패키지	AL2 최소	AL2023 최소
glibc-common	2.26	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
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-mod ules	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

패키지	AL2 최소	AL2023 최소
grub2-tools-minimal	2.06	2.06
grubby	8.28	8.40
gzip	1.5	1.12
hardlink	1.3	
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

패키지	AL2 최소	AL2023 최소
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
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

패키지	AL2 최소	AL2023 최소
libcroco	0.6.12	
libcrypt	2.26	
libcurl	8.3.0	
libcurl-minimal		8.5.0
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	

패키지	AL2 최소	AL2023 최소
libidn2	2.3.0	2.3.2
libkcapi		1.4.0
libkcapi-hmaccalc		1.4.0
libmetalink	0.1.3	
libmnl	1.0.3	1.0.4
libmodulemd		2.13.0
libmount	2.30.2	2.37.4
libnetfilter_conntrack	1.0.6	
libnfnetlink	1.0.1	
libnghttp2	1.41.0	1.59.0
libpcap	1.5.3	
libpipeline	1.2.3	1.5.3
libpng	1.5.13	
libpsl	0.21.5	0.21.1
libpwquality	1.2.3	1.4.4
librepo		1.14.5
libreport-filessystem		2.15.2
libseccomp	2.5.2	2.5.3
libselinux	2.5	3.4
libselinux-utils	2.5	3.4

패키지	AL2 최소	AL2023 최소
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	
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

패키지	AL2 최소	AL2023 최소
linux-firmware-whe nce		20210208(noarch)
logrotate	3.8.6	3.20.1
lua	5.1.4	
lua-libs		5.4.4
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	

패키지	AL2 최소	AL2023 최소
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	
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

패키지	AL2 최소	AL2023 최소
passwd	0.79	0.80
pciutils		3.7.0
pciutils-libs		3.7.0
pcre	8.32	
pcre2	10.23	10.40
pcre2-syntax		10.40
pinentry	0.8.1	
pkgconfig	0.27.1	
policycoreutils	2.5	3.4
popt	1.13	1.18
postfix	2.10.1	
procps-ng	3.3.10	3.3.17
psmisc	22.20	23.4
pth	2.0.7	
publicsuffix-list-dafsa	20240208	20240212
pygpgme	0.3	
pyliblzma	0.5.3	
python	2.7.18	
python2-cryptography	1.7.2	
python2-jsonschema	2.5.1	

패키지	AL2 최소	AL2023 최소
python2-oauthlib	2.0.1	
python2-pyasn1	0.1.9	
python2-rpm	4.11.3	
python2-setuptools	41.2.0	
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

패키지	AL2 최소	AL2023 최소
python3-docutils		0.16
python3-gpg		1.15.1
python3-hawkey		0.69.0
python3-idna		2.10
python3-jinja2		2.11.3
python3-jmespath		0.10.0
python3-jsonpatch		1.21
python3-jsonpointer		2.0
python3-jsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0
python3-libs		3.9.16
python3-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

패키지	AL2 최소	AL2023 최소
python3-prettytable		0.7.2
python3-prompt-toolkit		3.0.24
python3-pycparser		2.20
python3-pyrsistent		0.17.3
python3-pyserial		3.4
python3-pysocks		1.7.1
python3-pytz		2022.7.1
python3-pyyaml		5.4.1
python3-requests		2.25.1
python3-rpm		4.16.1.3
python3-ruamel-yaml		0.16.6
python3-ruamel-yaml-clib		0.1.2
python3-setuptools		4.4.1
python3-setuptools		59.6.0
python3-setuptools-wheel		59.6.0
python3-six		1.15.0
python3-systemd		235
python3-urllib3		1.25.10

패키지	AL2 최소	AL2023 최소
python3-wcwidth		0.2.5
python-babel	0.9.6	
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-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	

패키지	AL2 최소	AL2023 최소
python-pycurl	7.19.0	
python-repoze-lru	0.4	
python-requests	2.6.0	
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

패키지	AL2 최소	AL2023 최소
selinux-policy	3.13.1	38.1.45
selinux-policy-targeted	3.13.1	38.1.45
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

패키지	AL2 최소	AL2023 최소
tcp_wrappers-libs	7.6	
tzdata	2024a	2024a
update-motd	1.1.2	2.2
userspace-rcu		0.12.1
ustr	1.0.4	
util-linux	2.30.2	2.37.4
util-linux-core		2.37.4
vim-data	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
which	2.20	2.21
xfspgrog	5.0.0	5.18.0
xz	5.2.2	5.2.5
xz-libs	5.2.2	5.2.5
yum	3.4.3	4.14.0
yum-metadata-parser	1.1.4	
yum-plugin-priorities	1.1.31	
zlib	1.2.7	1.2.11
zram-generator		1.1.2
zram-generator-defaults		1.1.2

패키지	AL2 최소	AL2023 최소
zstd		1.5.5

Amazon Linux 2 및 Amazon Linux 2023 기본 컨테이너 이미지에 설치된 패키지 비교

Amazon Linux 2 및 AL2023 기본 컨테이너 이미지에 있는 RPM 간의 비교입니다.

패키지	AL2 컨테이너	AL2023 컨테이너
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	

패키지	AL2 컨테이너	AL2023 컨테이너
curl-minimal		8.5.0
cyrus-sasl-lib	2.1.26	
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	

패키지	AL2 컨테이너	AL2023 컨테이너
glibc-minimal-langpack	2.26	2.34
gmp	6.0.0	6.2.1
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	

패키지	AL2 컨테이너	AL2023 컨테이너
libcurl	8.3.0	
libcurl-minimal		8.5.0
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

패키지	AL2 컨테이너	AL2023 컨테이너
libsigsegv		2.13
libsmartcols		2.37.4
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

패키지	AL2 컨테이너	AL2023 컨테이너
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	
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

패키지	AL2 컨테이너	AL2023 컨테이너
pygpgme	0.3	
pyliblzma	0.5.3	
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

패키지	AL2 컨테이너	AL2023 컨테이너
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-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

AL1과 AL2023 비교

다음 주제에서는 [AL2와의 비교](#)에서 다루지 않은 AL1과 AL2023 간의 주요 차이점을 설명합니다.

Note

AL1은 2023년 12월 31일에 지원이 종료(EOL)되었으며, 2024년 1월 1일부터 보안 업데이트나 버그 수정을 받지 않습니다. AL1 EOL 및 유지 보수 지원에 대한 자세한 내용은 블로그 게시물 [Amazon Linux AMI 종료 업데이트](#)를 참조하세요. 애플리케이션을 AL2023으로 업그레이드하는 것이 좋습니다. 이 버전은 2028년까지 장기 지원을 포함합니다.

주제

- [각 릴리스에 대한 지원](#)
- [systemd는 upstart를 init 시스템으로 대체합니다.](#)
- [Python 2.6 및 2.7은 Python 3으로 대체되었습니다.](#)
- [OpenJDK 8은 가장 오래된 JDK](#)
- [AL2023 커널이 Amazon Linux 1\(AL1\)에서 변경](#)
- [Amazon Linux 1\(AL1\)과 Amazon Linux 2023 AMI에 설치된 패키지 비교](#)
- [Amazon Linux 1\(AL1\)과 Amazon Linux 2023 미니멀 AMI에 설치된 패키지 비교](#)
- [Amazon Linux 1\(AL1\)과 Amazon Linux 2023의 기본 컨테이너 이미지에 설치된 패키지 비교](#)

각 릴리스에 대한 지원

AL2023은 릴리스 날짜로부터 5년간 지원을 받습니다. AL1은 2020년 12월 31일부터 표준 지원을 종료하고 2023년 12월 31일부터 유지 관리 지원을 종료했습니다.

자세한 내용은 [릴리스 케이던스](#) 섹션을 참조하세요.

systemd는 upstart를 init 시스템으로 대체합니다.

AL2에서 systemd에 의해 upstart는 init 시스템으로 대체되었습니다. 또한 AL2023은 init 시스템으로 systemd를 사용하여 systemd의 새로운 기능과 역할을 추가했습니다.

Python 2.6 및 2.7은 Python 3으로 대체되었습니다.

AL1은 2018.03 릴리스에서 Python 2.6을 EOL로 표시했지만 패키지를 설치할 리포지토리에서 계속 사용할 수 있었습니다. AL2는 지원되는 가장 빠른 Python 버전으로 Python 2.7과 함께 제공되었으며 AL2023은 Python 3으로의 전환을 완료합니다. AL2023 리포지토리에는 Python 2.x 버전이 포함되지 않습니다.

Amazon Linux의 Python에 대한 자세한 내용은 [AL2023에서 Python 사용](#)을 참조하십시오.

OpenJDK 8은 가장 오래된 JDK

AL2023은 기본 (그리고 유일한) Java 개발 키트(JDK)로 [Amazon Corretto](#)를 지원합니다. AL2023에 설치된 모든 Java 기반 패키지는 모두 Amazon Corretto 17로 빌드됩니다.

AL1에서 OpenJDK 1.6.0(java-1.6.0-openjdk)은 첫 번째 2018.03 릴리스를 끝으로 EOL되었으며, OpenJDK 1.7.0(java-1.7.0-openjdk)은 2020년 중반에 EOL되었습니다. 하지만 AL1 리포지토리에서 두 버전 모두 사용할 수 있었습니다. AL2023에서 실행할 수 있는 가장 오래된 OpenJDK 버전은 Amazon Corretto 8에서 제공하는 OpenJDK 8입니다.

AL2023 커널이 Amazon Linux 1(AL1)에서 변경

커널 라이브 패치

AL2023과 AL2 모두 커널 라이브 패치 기능에 대한 지원을 추가합니다. 이를 통해 재부팅이나 가동 중지 시간 없이 Linux 커널의 치명적이고 심각한 보안 취약성을 패치할 수 있습니다. 자세한 내용은 [AL2023의 커널 라이브 패치](#) 단원을 참조하십시오.

커널 파일 시스템 지원

AL1 커널에서 마운트를 지원하는 파일 시스템이 몇 가지 변경되었으며 구문 분석할 파티션 스키마도 변경되었습니다.

CONFIG 옵션	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_AIS_FS	m	n	n	n	n	n	n

CONFIG 옵션	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_AIO_RRPC	m	n	n	n	n	n	n
CONFIG_BLOCK_DISKLABEL	y	n	n	n	n	n	n
CONFIG_CIFS_AMFS	m	n	n	n	n	n	n
CONFIG_CIFS_AMFS_BLOCKDEV	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_DLM_CLONE	해당 사항 없음	n	n	n	n	n	n
CONFIG_DLM_ERA	n	n	n	n	n	n	n
CONFIG_DM_INTEGRITY	m	m	m	m	m	m	m
CONFIG_DM_LOG_WRITES	n	m	m	m	m	m	m
CONFIG_DM_SWITCH	n	n	n	n	n	n	n
CONFIG_DM_VERITY	n	m	m	m	m	m	m

CONFIG 옵션	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_CRYPTO_FS	m	n	n	n	n	n	n
CONFIG_EXT2_FS	해당 사항 없음	m	m	m	m	m	m
CONFIG_EXT3_FS	m	n	n	n	n	n	n
CONFIG_EXT4_FS	m	n	n	n	n	해당 사항 없음	해당 사항 없음
CONFIG_GFS2_FS	n	n	n	n	n	n	n
CONFIG_HFSPLUS_FS	m	n	n	n	n	n	n
CONFIG_HFS_FS	m	n	n	n	n	n	n
CONFIG_JFS_FS	n	n	n	n	n	n	n
CONFIG_LVM2_METADATA	y	n	n	n	n	n	n
CONFIG_MMC_PARTITION	y	n	n	n	n	n	n
CONFIG_NFS_V2	m	n	n	n	n	n	n

CONFIG 옵션	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_NFS_FS	m	n	n	n	n	n	n
CONFIG_ROMFS_FS	m	n	n	n	n	n	n
CONFIG_SCLARIS_X86_PARTITION	y	n	n	n	n	n	n
CONFIG_SQUASHFS_ZSTD	y	y	y	y	y	y	y
CONFIG_SUN_PARTITION	y	n	n	n	n	n	n

보안에 중점을 둔 커널 구성 변경

CONFIG 옵션	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_BUG_ON_DATA_CORRUPTION	y	y	y	y	y	y	y
CONFIG_DEBUG_FAULT_MMIO	4096	65536	65536	65536	65536	65536	65536

CONFIG 옵션	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
P_MIN_ADJ_R							
CONFIG_DM_VMEM	y	n	n	n	n	n	n
CONFIG_DM_VPORT	y	n	n	n	n	n	n
CONFIG_FORTIFY_SOURCE	y	y	y	y	y	y	y
CONFIG_HARDENED_USERSPACE_COPYING	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_INIT_ON_ALLOC_DEFAULT_ON	해당 사항 없음	n	n	n	n	n	n
CONFIG_INIT_ON_FREE_DEFAULT_ON	해당 사항 없음	n	n	n	n	n	n
CONFIG_IOMMU_DEFAULT_DMA_SRICT	해당 사항 없음	n	n	n	n	n	n

CONFIG 옵션	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	해당 사항 없음	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	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_SLAB_Linux_Disable	y	n	n	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_SWAPFILE_PAGE_ALLOCATION	해당 사항 없음	y	y	y	y	y	y
CONFIG_SWAP_FREEZE_HARDENED	y	y	y	y	y	y	y

CONFIG 옵션	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_SLAB_FREELIST_RANDOM	n	y	y	y	y	y	y

기타 커널 구성 변경

CONFIG 옵션	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_NUM_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	해당 사항 없음	해당 사항 없음
CONFIG_PRINTK	m	m	m	m	m	m	m
CONFIG_SMP	m	n	n	n	n	n	n
CONFIG_X86_PV	y	해당 사항 없음	n	해당 사항 없음	n	N/A	n

Amazon Linux 1(AL1) 과 Amazon Linux 2023 AMI에 설치된 패키지 비교

AL1 및 AL2023 표준 AMI에 있는 RPM 간의 비교입니다.

패키지	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	

패키지	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

패키지	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

패키지	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	

패키지	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

패키지	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

패키지	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

패키지	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	

패키지	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

패키지	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

패키지	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

패키지	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

패키지	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

패키지	AL1 AMI	AL2023 AMI
libnetfilter_contrack	1.0.4	
libnfnetlink	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-filessystem		2.15.2
libseccomp		2.5.3
libselinux	2.1.10	3.4
libselinux-utils	2.1.10	3.4

패키지	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

패키지	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

패키지	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	

패키지	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

패키지	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

패키지	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

패키지	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-I/O		1.43

패키지	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

패키지	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

패키지	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	

패키지	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	

패키지	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

패키지	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-jsonschema		3.2.0
python3-libcomps		0.1.20
python3-libdnf		0.69.0

패키지	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

패키지	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

패키지	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	

패키지	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

패키지	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

패키지	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

패키지	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

패키지	AL1 AMI	AL2023 AMI
zstd		1.5.5

Amazon Linux 1(AL1)과 Amazon Linux 2023 미니멀 AMI에 설치된 패키지 비교

AL1 및 AL2023 최소 AMI에 있는 RPM 간의 비교입니다.

패키지	AL1 최소	AL2023 최소
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	

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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	

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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
libcgroupp	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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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
libselinux	2.1.10	3.4
libselinux-utils	2.1.10	3.4
libsemanage	2.1.6	3.4
libsepol	2.1.7	3.4
libsigsegv		2.13
libsmartcols	2.23.2	2.37.4
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	

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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	

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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

패키지	AL1 최소	AL2023 최소
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	

패키지	AL1 최소	AL2023 최소
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	

패키지	AL1 최소	AL2023 최소
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) 과 Amazon Linux 2023의 기본 컨테이너 이미지에 설치된 패키지 비교

AL1 및 AL2023 기본 컨테이너 이미지에 있는 RPM 간의 비교입니다.

패키지	AL1 컨테이너	AL2023 컨테이너
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	

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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

패키지	AL1 컨테이너	AL2023 컨테이너
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 시스템 요구 사항

이 섹션에서는 AL2023 사용에 대한 시스템 요구 사항을 설명합니다.

주제

- [AL2023 실행을 위한 CPU 요구 사항](#)
- [AL2023 실행을 위한 메모리\(RAM\) 요구 사항](#)

AL2023 실행을 위한 CPU 요구 사항

AL2023 코드를 실행하려면 사용되는 프로세서가 특정 최소 요구 사항을 충족해야 합니다. 이러한 요구 사항을 충족하지 않는 CPU에서 AL2023 실행을 시도하면 코드 실행 초기에 잘못된 명령 오류가 발생할 수 있습니다.

최소 요구 사항은 [Amazon EC2 AL2023](#), [AL2023 컨테이너](#) 및 [Amazon EC2 외부 AL2023](#)에 적용됩니다.

AL2023 ARM CPU 사양

모든 AL2023 aarch64(ARM) 바이너리는 64비트용으로 빌드됩니다. 32비트 ARM 바이너리를 사용할 수 없으므로 64비트 ARM CPU가 필요합니다.

Note

ARM 기반 인스턴스에서 AL2023은 Graviton2 이상 프로세서를 사용하는 인스턴스 유형만 지원합니다. AL2023은 A1 인스턴스를 지원하지 않습니다.

AL2023을 사용하려면 암호화 확장(ARMv8.2+crypto)을 갖춘 ARMv8.2 호환 프로세서가 필요합니다. aarch64에 대한 모든 AL2023 패키지는 -march=armv8.2-a+crypto 컴파일러 플래그로 빌드됩니다. 구형 ARM 프로세서에서 AL2023 코드를 실행하려고 할 때 정상적인 오류 메시지를 출력하려고 시도하지만, 첫 번째 오류 메시지가 잘못된 명령 오류일 수 있습니다.

Note

AL2023 aarch64 기본 CPU 요구 사항으로 인해 Raspberry Pi 5 이전의 모든 Raspberry Pi 시스템은 최소 CPU 요구 사항을 충족하지 않습니다.

AL2023 x86-64 CPU 사양

모든 AL2023 x86-64 바이너리는 컴파일러에 `-march=x86-64-v2`를 전달하여 x86-64 아키텍처의 x86-64v2 개정에 맞게 빌드됩니다.

아키텍처의 x86-64v2 개정은 기본 x86-64 아키텍처에 다음 CPU 기능을 추가합니다.

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

이는 대략 2009년 이후에 출시된 x86-64 프로세서에 매핑됩니다. 예를 들어 Intel Nehalem, AMD Jaguar, Atom Silvermont와 함께 VIA Nano 및 Eden C 마이크로아키텍처가 있습니다.

Amazon EC2의 모든 x86-64 인스턴스 유형은 M1, C1 및 M2 인스턴스 패밀리 등 x86-64v2를 지원합니다.

32비트 x86(i686) AL2023 바이너리는 빌드되지 않습니다. AL2023은 32비트 사용자 공간 바이너리 실행을 계속 지원하지만 이 기능은 더 이상 사용되지 않으며 향후 주요 Amazon Linux 버전에서 제거될 수 있습니다. 자세한 내용은 [32비트 x86 \(i686\) 패키지](#) 섹션을 참조하세요.

AL2023 실행을 위한 메모리(RAM) 요구 사항

Amazon EC2 .nano 인스턴스 유형 제품군(t2.nano, t3.nano, t3a.nano 및 t4g.nano)의 RAM은 512MB이며, 이는 AL2023의 최소 요구 사항입니다.

Note

최소 요구 사항은 512MB이지만 이러한 인스턴스 유형은 메모리 제약이 있으며 기능과 성능이 제한될 수 있습니다.

AL2023 이미지는 RAM이 512MB 미만인 시스템에서 테스트되지 않았습니다. 512MB 미만의 RAM으로 AL2023 기반 컨테이너 이미지를 실행하는 것은 컨테이너화된 워크로드에 따라 달라집니다.

일부 AL2023 릴리스 간 `dnf upgrade` 같은 일부 워크로드는 512MB 이상의 RAM이 필요할 수 있습니다. 이러한 이유로 [AL2023.3](#) 릴리스에는 RAM이 800MB 미만인 인스턴스에 대해 기본적으로 `zram` 활성화가 도입되었습니다. 컨테이너화된 워크로드의 경우 이 정도의 메모리를 사용하는 AL2023 인스턴스에서는 일부 워크로드가 정상적으로 실행되지만 메모리 사용량이 이 정도로 제한된 컨테이너에서 실행하면 실패할 수 있습니다.

RAM이 800MB 미만인 인스턴스 유형의 경우 AL2023 ([AL2023.3](#) 이상) 는 기본적으로 `zram` 기반 스왑을 활성화합니다. 메모리가 800MB 미만인 Amazon EC2 인스턴스 유형으로 `t4g.nano`, `t3a.nano`, `t3.nano`, `t2.nano`, `t1.micro` 등이 있습니다. 즉, AL2023 명령으로 메모리 페이지를 온디맨드 방식으로 압축 및 압축 해제하므로 이러한 인스턴스 유형의 메모리 부족 현상이 줄어듭니다. 이렇게 하면 압축을 수행하는 데 필요한 CPU 사용량을 희생시키면서 더 많은 메모리가 있는 인스턴스 유형이 필요한 워크로드도 가능해집니다.

Amazon Linux 커널

Amazon Linux 2023(AL2023)은 업스트림 Linux 커뮤니티의 연간 LTS 커널을 기반으로 매년 Q1에 새로운 장기 지원(LTS) 커널을 릴리스합니다. 각각의 새로운 LTS 커널은 업스트림 Linux 커널의 최신 보안 수정 사항, 성능 개선 사항, 하드웨어 지원 및 기능을 제공합니다.

커널 수명 주기

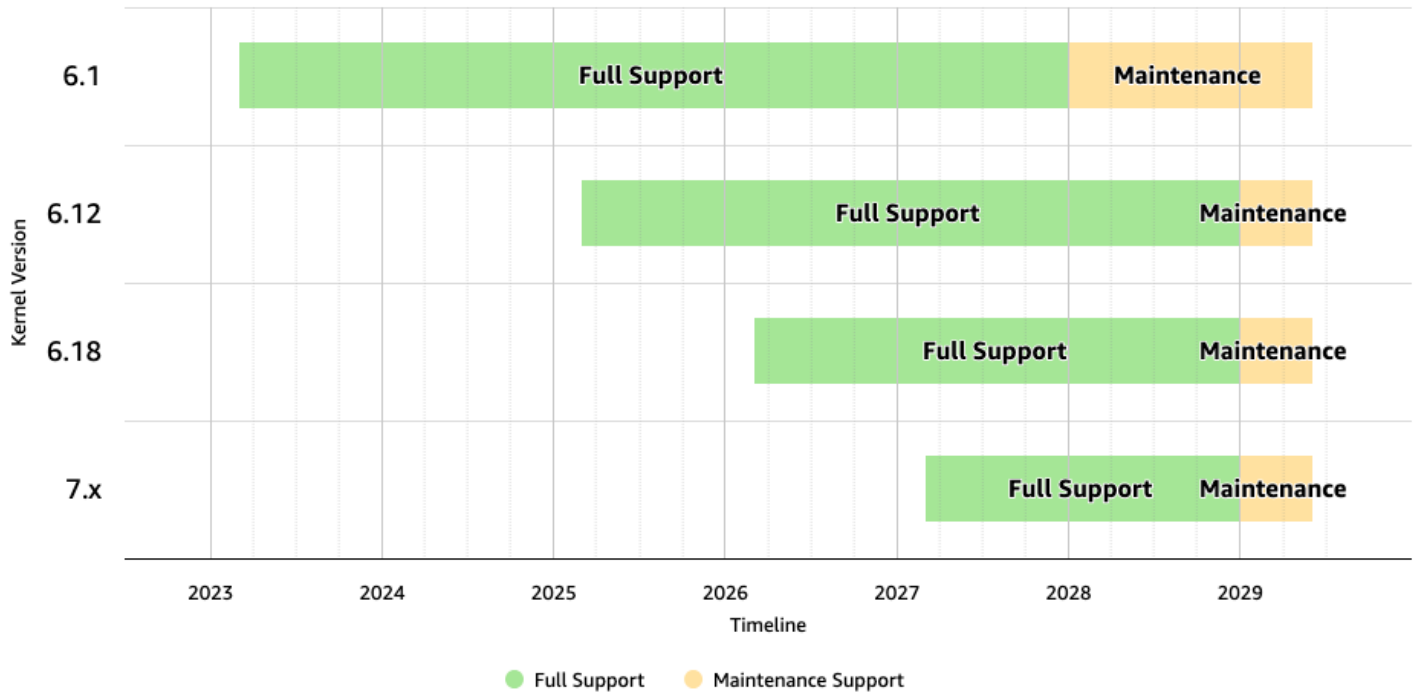
각 AL2023 LTS 커널은 2단계로 4년 동안 지원됩니다.

1. 전체 지원(1~2년) - 커널은 업스트림 LTS 커널의 정기적인 리베이스를 통해 모든 CVE 심각도에 대한 수정 사항을 수신합니다. 이 단계는 업스트림 Linux 커뮤니티의 LTS 지원 기간과 일치합니다. 업스트림이 LTS 지원 기간을 확장하는 경우 Amazon Linux가 적합합니다.
2. 유지 관리 지원(3~4년) - 업스트림 LTS 지원 기간이 종료된 후에도 Amazon Linux 팀은 주로 중요 및 중요 CVEs(CVSS 점수 7.0 이상)와 알려진 악용된 취약성에 대한 수정 사항을 계속 백포트합니다. 이 단계에서는 심각도 CVEs가 백포트되지 않습니다.

4년이 지나면 커널의 수명이 종료되어 보안 업데이트가 더 이상 수신되지 않습니다. 이전 커널은 리포지토리를 통해 계속 사용할 수 있으며 계속 사용할 수 있습니다. 추가 패치나 수정 사항을 기대해서는 안 됩니다. 이 날짜 이전에 지원되는 커널로 업그레이드하는 것이 좋습니다. 커널별 AMIs 커널 지원 종료일 이후에는 더 이상 업데이트되지 않습니다.

다음 차트는 현재 Amazon Linux LTS 커널에 대한 지원 타임라인을 보여줍니다.

Amazon Linux 2023 Kernel Release Timeline



커널 업데이트

2026년 6월부터 AL2023은 기본 커널을 매년 업데이트합니다. AMIs [al2023-ami-kernel-default](#) 세트는 최신 LTS 커널로 업데이트되므로 새로 시작된 인스턴스에 새 커널 버전이 제공됩니다.

실행 중인 인스턴스는 새 커널로 자동 업데이트되지 않습니다. 기존 인스턴스에서 커널을 업그레이드하려면 새 커널 패키지를 명시적으로 설치하고 재부팅해야 합니다. 자세한 내용은 [AL2023에서 Linux 커널 업데이트](#)를 참조하세요.

최신 보안 및 성능 개선의 이점을 활용하려면 새 커널을 즉시 채택하는 것이 좋습니다. 오래된 커널은 시간이 지남에 따라 보안 수정이 더 적고 느려집니다. 커널 업데이트를 기존 테스트 및 배포 파이프라인에 통합하여 프로덕션으로 롤아웃하기 전에 호환성을 검증합니다.

배포 간 커널 버전 중첩

Amazon Linux 배포 간 마이그레이션을 간소화하기 위해 현재 및 다음 Amazon Linux 배포 모두에서 하나 이상의 커널 버전을 사용할 수 있습니다. 이렇게 하면 먼저 기존 배포에서 새 커널로 업그레이드하고 워크로드를 검증한 다음 동일한 커널 버전을 사용할 수 있다는 확신을 갖고 새 배포로 마이그레이션할 수 있습니다.

CVE 처리

AL2023은 다음과 같이 커널 CVEs 처리합니다.

- 중요하고 중요한 CVEs(CVSS 7.0 이상) 및 알려진 악용된 취약성은 지원되는 모든 커널로 백포트됩니다.
- 낮음 및 중간 CVEs(CVSS 7.0 미만)는 전체 지원 단계에서 일반 업스트림 LTS 리베이스를 통해 처리됩니다. 유지 관리 지원 단계에서는 이러한 CVEs 백포트되지 않습니다. 업스트림 LTS 리베이스에서 60일 이내에 낮은 CVE 또는 중간 CVE가 해결되지 않으면 [Amazon Linux 보안 센터에서](#) "계획된 수정 사항 없음"으로 표시됩니다.

사용 가능한 최신 커널을 실행하는 것이 최신 보안, 성능 및 기능 업데이트를 가져오는 가장 좋은 방법입니다.

해야 할 일

1. 애플리케이션에 지속적 통합(CI) 구현 - 프로덕션 설정을 변경하기 전에 테스트 단계에서 제대로 검증되었는지 확인합니다. 검증에 커널 업데이트를 포함합니다.
2. 최신 커널 유지 - 사용 가능한 각 새 연간 LTS 커널을 즉시 채택합니다. 최신 커널은 보안 수정을 더 빠르게 수신합니다. [al2023-ami-kernel-default](#) 일련의 AMIs가 Amazon Linux 팀에서 권장하는 커널 버전에 자동으로 있다고 가정합니다.
3. 업데이트 자동화 - [AWS Systems Manager](#)와 같은 도구를 사용하여 플릿 전체에서 커널 업데이트를 관리합니다.
4. 재부팅 계획 - 모든 커널 업데이트에는 재부팅이 필요합니다. 유지 관리 일정에 재부팅 기간을 구축합니다.

AL2023 그래픽 데스크톱

Amazon Linux 2023은 릴리스 2023.7부터 GNOME을 기반으로 선택적인 경량 클라우드 최적화 그래픽 인터페이스를 제공합니다. 이 최신 데스크톱 환경은 안전한 브라우징을 위한 Firefox와 같은 내장 도구를 통해 향상된 생산성 기능을 제공하는 동시에 원격 액세스 시 Amazon DCV 및 VNC 지원을 유지합니다.

Note

AL2023은 업스트림 Firefox 추가 지원 릴리스(ESR) 버전을 면밀히 따르고 가능한 한 빨리 다음 ESR로 업데이트합니다. 자세한 내용은 [Firefox ESR 릴리스 일정](#) 및 [Firefox 릴리스 정보를](#) 참조하세요.

관련 주제

그래픽 데스크톱 환경 설치에 대한 자세한 내용은 다음 설명서를 참조하세요.

- [자습서: AL2023에 GNOME 데스크톱 환경 설치](#)

Amazon Linux용 보조 패키지

이 섹션에서는 Amazon Linux용 보조 패키지(SPAL)를 소개하고 패키지 관련 문제를 보고하기 위한 지침과 함께 이점 및 제한 사항을 간략하게 설명합니다.

주제

- [Amazon Linux\(또는 SPAL\)용 보조 패키지란 무엇입니까?](#)
- [이점](#)
- [SPAL 패키지 지원](#)
- [패키지 관련 문제 보고](#)
- [관련 주제](#)
- [자습서: AL2023에서 SPAL 리포지토리 구성](#)

Amazon Linux(또는 SPAL)용 보조 패키지란 무엇입니까?

Amazon Linux용 보조 패키지(SPAL)는 [Enterprise Linux 9용 추가 패키지\(EPEL9\)에서 파생된 수천 개의 추가 패키지에 대한 액세스를 제공하는 전용 패키지 리포지토리입니다](#). 이러한 패키지는 코어 Amazon Linux 2023에서 사용할 수 있는 기존 소프트웨어를 보완합니다.

SPAL은 AL2023과 호환되는 사전 빌드된 패키지를 제공하여 소프트웨어 배포를 간소화하므로 고객이 소스 코드 자체에서 패키지를 빌드할 필요가 없습니다. 이렇게 하면 소프트웨어 설치 프로세스에서 시간과 노력을 절약할 수 있습니다.

Note

SPAL은 릴리스 버전이 2023.9.20251117 이상인 AL2023 인스턴스의 AWS경우 GovCloud(미국) 리전 및 중국을 포함한 모든 AWS상용 리전에서 사용할 수 있습니다.

이점

SPAL은 Amazon Linux 2023 사용자에게 다음과 같은 몇 가지 주요 이점을 제공합니다.

- AL2023 사용 사례 확장 - pandoc GDAL 또는와 같은 핵심 AL2023 리포지토리 이외의 인기 있는 추가 패키지에 액세스drbd-utils하면 고객이 여러 비즈니스 및 개발 요구 사항을 지원할 수 있습니다.

- AL2023 패키지 관리 간소화 - AL2023용으로 사전 구축된 패키지를 제공하면 소스에서 추가 패키지를 빌드하는 프로세스가 제거되므로 시간을 절약하고 컴파일 오류 위험을 줄일 수 있습니다.
- AL2에서 AL2023으로 마이그레이션 간소화 - SPAL 리포지토리를 사용하면 EPEL7 패키지에 의존하는 워크로드를 포함하여 AL2에서 AL2023으로 워크로드를 원활하게 마이그레이션할 수 있습니다.

SPAL 패키지 지원

SPAL 패키지에는 Amazon Linux 2023의 전체 수명 동안 지원을 받는 코어 AL2023 패키지와 동일한 수준의 지원이 적용되지 않습니다.

Important

SPAL을 사용하기 전에 고객은 다음 고려 사항을 신중하게 평가해야 합니다.

- SPAL 패키지는 지원 플랜에서 다루지 않습니다.AWS
- SPAL 패키지는 업스트림 EPEL9에서 '있는 그대로' 제공됩니다. EPEL9
- SPAL 패키지는 CVE 보안 추적을 받지 않습니다AWS.
- SPAL 패키지는 사용 가능한 경우 업스트림 EPEL9에서만 보안 패치 및 버그 수정을 수신합니다.

패키지 관련 문제 보고

SPAL 패키지에 문제가 발생하면 먼저 업스트림 EPEL9 리포지토리의 해당 패키지에서 동일한 문제가 발생하는지 확인하는 것이 좋습니다. 이를 위해 업스트림 EPEL 리포지토리 [의 문제 목록](#)을 참조하세요.

EPEL9에 문제가 없는 경우 [Amazon Linux 2023 GitHub 리포지토리](#)에서 문제를 생성합니다. 이는 문제가 SPAL 패키지 빌드 또는 구성에 고유함을 나타냅니다.

이 접근 방식은 적절한 유지 관리 담당자가 문제를 해결하고 SPAL 및 업스트림 EPEL 패키지의 전반적인 품질에 기여하도록 합니다.

Note

보고된 문제는 최선을 다해 처리됩니다.

관련 주제

시스템에서 SPAL을 구성하는 방법에 대한 자세한 내용은 다음 설명서 페이지를 참조하세요.

- [자습서: AL2023에서 SPAL 리포지토리 구성](#)

자습서: AL2023에서 SPAL 리포지토리 구성

Amazon Linux용 보조 패키지(SPAL)는 고객에게 수천 개의 오픈 소스 패키지에 대한 액세스를 제공하는 AL2023용 추가 패키지 리포지토리입니다.

다음 자습서는 AL2023 인스턴스에서 SPAL 리포지토리를 구성하는 데 도움이 됩니다. 리포지토리를 설치하면 SPAL에서 사용할 수 있는 모든 RPM 패키지에 액세스할 수 있습니다. 설치되면 패키지 관리자 사용하여 시스템에 이러한 패키지를 설치하고 사용할 수 있습니다.

내용

- [사전 조건](#)
- [사전 조건 확인](#)
- [시스템에 SPAL 설치](#)
- [SPAL 패키지 설치](#)
- [시스템에서 SPAL 리포지토리 제거](#)
- [관련 주제](#)

사전 조건

이 자습서에서는 AL2023 릴리스 버전 2023.9.20251117 이상을 사용하여 인스턴스를 이미 시작했다고 가정합니다. 자세한 내용은 [Amazon EC2 AL2023](#) 및 [AL2023 업데이트](#) 페이지를 참조하세요.

사전 조건 확인

- 인스턴스가 사전 조건을 충족하는지 확인하려면 시스템에 system-release 설치된 버전을 확인할 수 있습니다.

패키지 버전을 확인하려면 다음 명령을 사용할 수 있습니다.

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

명령은 메이저 버전을 포함하여 패키지에 대한 정보를 표시합니다.

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

Note

최신 버전의가 system-release 설치되어 있는지 확인합니다. 를 실행sudo dnf upgrade하여 최신 버전으로 업데이트할 수 있습니다.

시스템에 SPAL 설치

1. 시스템에 spal-release 패키지를 설치합니다. 그러면 .repo 구성 파일과 GPG 키가 시스템에 추가됩니다.

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

Note

설치 중에 지원 문이 표시됩니다. 이 문은 SPAL의 지원 및 제한 범위를 설명합니다. 시간을 내어이 정보를 주의 깊게 검토하세요.

2. SPAL 리포지토리 구성이 시스템에 성공적으로 추가되었는지 확인합니다.

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

시스템에 구성된 두 개의 리포지토리amazonlinux-spal가 표시됩니다. amazonlinux-spal-source

를 실행하여 구성된 리포지토리 목록을 확인할 수도 있습니다dnf repolist.

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

Note

--all 플래그는 활성화된 리포지토리와 비활성화된 리포지토리를 모두 보는 데 필요합니다.

두 SPAL 리포지토리를 모두 사용할 수 있어야 합니다. Amazon Linux 2023 SPAL 리포지토리 - 소스 패키지 리포지토리는 기본적으로 비활성화되어 있습니다.

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

3. (선택 사항) 소스 리포지토리를 활성화합니다.

Note

RPM 소스(SRPM) 리포지토리는 일반적으로 개발자가 소프트웨어 설치를 위해 최종 사용자가 아닌 패키지를 빌드하는 데 주로 사용되므로 기본적으로 비활성화됩니다. DNF는 와 같은 소스 패키지가 필요한 명령을 사용할 때 소스 리포지토리를 자동으로 활성화합니다 `dnf download --source package`.

일회성 소스 패키지 작업을 위해 소스 리포지토리를 수동으로 활성화할 필요가 없습니다. 시스템의 SPAL에서 SRPMs 다시 빌드하려는 경우에만 이 단계를 따르세요.

시스템에서 Amazon Linux 2023 SPAL 리포지토리 - 소스 패키지 리포지토리를 영구적으로 활성화하려면 다음 명령을 실행합니다.

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

SPAL 패키지 설치

- `dnf install` 명령을 실행하여 시스템에 SPAL 패키지를 설치합니다.


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

Note

dnf list를 사용하여 SPAL 패키지의 전체 목록을 볼 수 있습니다.

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

Note

SPAL은 버전이 지정된 리포지토리입니다. 최신 패키지 목록을 보려면 최신 버전의 system-release 설치되어 있어야 합니다.

결정적 업데이트에 대한 자세한 내용은 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#)

시스템에서 SPAL 리포지토리 제거

1. dnf remove 명령을 사용하여 SPAL 리포지토리 구성을 제거합니다.

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

2. dnf repolist 명령을 실행하여 리포지토리가 제거되었는지 확인합니다.

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

Important

시스템에서 SPAL 리포지토리 구성을 제거해도 시스템에 설치된 SPAL 패키지는 제거되지 않습니다.

관련 주제

Amazon Linux용 추가 패키지 리포지토리에 대한 자세한 내용은 다음 설명서를 참조하세요.

- [Amazon Linux용 보조 패키지](#)

AL2023에서 애플리케이션 실행

이 섹션에서는 Amazon Linux 2023(AL2023)에서 애플리케이션을 실행하는 방법을 다룹니다. 여기에는 애플리케이션이 시작(다시 시작)되는 시점 관리 및 리소스 사용량 제어가 포함됩니다.

주제

- [systemd를 사용하여 AL2023에서 프로세스 리소스 사용 제한](#)
- [cgroups를 사용하여 AL2023에서 프로세스 리소스 사용 제한](#)

systemd를 사용하여 AL2023에서 프로세스 리소스 사용 제한

Amazon Linux 2023(AL2023)에서는 systemd를 사용하여 프로세스 또는 프로세스 그룹에서 사용할 수 있는 리소스를 제어하는 것이 좋습니다. systemd 사용은 cgroups를 수동으로 조작하거나, 이전에는 타사 [EPEL](#) 리포지토리의 Amazon Linux에서만 사용할 수 있었던 [cpulimit](#) 같은 유틸리티의 사용을 대체할 수 있는 강력하고 간편한 방법입니다.

자세한 내용은 [systemd.resource-control](#)에 대한 업스트림 systemd 설명서 또는 AL2023 인스턴스의 `systemd.resource-control`에 대한 man 페이지를 참조하세요.

아래 예제에서는 stress-ng CPU 스트레스 테스트(stress-ng 패키지)를 사용하여 CPU 사용량이 많은 애플리케이션을 시뮬레이션하고, memcached를 사용하여 메모리 사용량이 많은 애플리케이션을 시뮬레이션합니다.

아래 예제에서는 일회성 명령에 CPU 제한을 적용하고 서비스에 메모리 제한을 적용하는 방법을 다룹니다. systemd에서 제공하는 대부분의 리소스 제약 조건은 systemd가 프로세스를 실행할 모든 위치에서 사용할 수 있으며 여러 개를 동시에 사용할 수 있습니다. 아래 예제는 설명을 위한 단일 제약으로 제한됩니다.

일회성 명령 실행을 위한 **systemd-run**을 사용한 리소스 제어

일반적으로 시스템 서비스와 연결되지만 systemd는 루트 사용자가 아닌 사용자가 서비스를 실행하거나 타이머를 예약하거나 일회성 프로세스를 실행하는 데도 사용할 수 있습니다. 다음 예제에서는 stress-ng를 예제 애플리케이션으로 사용합니다. 첫 번째 예제에서는 ec2-user 기본 계정에서 systemd-run을 사용하여 실행하고, 두 번째 예제에서는 CPU 사용량에 제한을 둡니다.

Example명령줄에서 systemd-run을 사용하여 리소스 사용량을 제한하지 않고 프로세스를 실행합니다.

1. stress-ng 패키지를 예제에 사용할 예정이므로 해당 패키지가 설치되어 있는지 확인합니다.

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

2. 사용 가능한 CPU 양을 제한하지 않고 systemd-run으로 10초 CPU 스트레스 테스트를 실행합니다.

```
[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 옵션은 systemd-run에 로그인한 사용자로 명령을 실행하도록 지시하고, --tty 옵션은 TTY가 연결되어 있음을 의미하며, --wait는 서비스가 완료될 때까지 기다리는 것을 의미하고, --property=CPUAccounting=1 옵션은 systemd-run에 프로세스를 실행하는 데 사용되는 CPU 시간을 기록하도록 지시합니다. --property 명령줄 옵션을 사용하여 systemd.unit 구성 파일에서 구성할 수 있는 systemd-run 설정을 전달할 수 있습니다.

CPU에 로드를 배치하라는 지시를 받으면 stress-ng 프로그램은 실행을 요청하는 기간 동안 사용 가능한 모든 CPU 시간을 사용하여 테스트를 수행합니다. 실제 애플리케이션의 경우 프로세스의 총 런타임에 제한을 두는 것이 좋습니다. 아래 예제에서는 systemd-run을 사용하여 최대 기간 제한보다 더 오래 stress-ng를 실행하도록 요청합니다.

Example명령줄에서 systemd-run을 사용하여 프로세스를 실행하고 CPU 사용량을 1초로 제한합니다.

1. 이 예제를 실행하려면 stress-ng가 설치되어 있는지 확인합니다.
2. LimitCPU 속성은 이 프로세스가 사용할 수 있는 CPU의 최대 시간을 제한하는 ulimit -t와 동일합니다. 이 경우 10초 스트레스 실행을 요청하고 CPU 사용량을 1초로 제한하므로 명령은 SIGXCPU 신호를 수신하고 실패합니다.

```
[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
```

더 일반적으로는 특정 프로세스에서 사용할 수 있는 CPU 시간 비율을 제한할 수 있습니다. 아래 예제에서는 stress-ng에서 사용할 수 있는 CPU 시간의 비율을 제한합니다. 실제 서비스의 경우 사용자 요청을 처리하는 프로세스에 리소스를 자유롭게 두기 위해 백그라운드 프로세스에서 사용할 수 있는 최대 CPU 시간 비율을 제한하는 것이 좋습니다.

Example하나의 CPU에서 프로세스를 CPU 시간의 10%로 제한하는 데 systemd-run 사용

1. 이 예제를 실행하려면 stress-ng가 설치되어 있는지 확인합니다.
2. CPUQuota 속성을 사용하여 systemd-run에 실행할 명령의 CPU 사용량을 제한하도록 지시합니다. 프로세스가 실행할 수 있는 시간만 제한하는 것이 아니라 사용할 수 있는 CPU 양을 제한합니다.

```
[ec2-user ~]$ systemd-run --user --tty --wait --property=CPUAcounting=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 회계를 통해 서비스가 10초 동안 실행되었지만 실제 CPU 시간 중 1초만 소비되었음을 알 수 있습니다.

CPU, 메모리, 네트워킹 및 IO에 대한 리소스 사용을 제한하도록 systemd를 구성하는 방법에는 여러 가지가 있습니다. 포괄적인 설명서는 [systemd.resource-control](#)에 대한 업스트림 systemd 설명서 또는 AL2023 인스턴스의 `systemd.resource-control`에 대한 man 페이지를 참조하세요.

백그라운드에서 systemd는 cgroups 같은 Linux 커널의 기능을 사용하여 이러한 제한을 구현하는 동시에 수동으로 구성할 필요가 없도록 합니다. [cgroup-v2용 Linux 커널 설명서](#)에는 cgroups 작동 방식에 대한 광범위한 세부 정보가 포함되어 있습니다.

systemd 서비스의 리소스 제어

시스템 리소스 사용을 제어하기 위해 systemd 서비스의 [Service] 섹션에 추가할 수 있는 몇 가지 파라미터가 있습니다. 여기에는 하드 제한과 소프트 제한이 모두 포함됩니다. 각 옵션의 정확한 동작은 [systemd.resource-control](#)에 대한 업스트림 systemd 설명서 또는 AL2023 인스턴스의 `systemd.resource-control`에 대한 man 페이지를 참조하세요.

일반적으로 사용되는 제한은 메모리 사용량에 대한 스로틀링 제한을 지정하는 MemoryHigh, 하드 상한(이에 도달하면 OOM 킬러가 간접 호출됨)을 설정하는 MemoryMax 및 CPUQuota(이전 섹션에서 설명)입니다. 고정된 숫자 대신 가중치와 우선순위를 구성할 수도 있습니다.

Example systemd를 사용하여 서비스에 대한 메모리 사용량 제한 설정

이 예제에서는 간단한 키-값 캐시인 memcached에 대한 하드 메모리 사용량 제한을 설정하고 전체 시스템이 아닌 해당 서비스에 대해 OOM 킬러가 간접 호출되는 방법을 보여줍니다.

1. 먼저 이 예제에 필요한 패키지를 설치해야 합니다.

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

2. memcached.service를 활성화한 다음 memcached가 실행되도록 서비스를 시작합니다.

```
[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.service가 실행 중인지 확인합니다.

```
[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. memcached가 설치되고 실행 중이므로 일부 무작위 데이터를 캐시에 삽입하여 작동하는지 확인할 수 있습니다.

/etc/sysconfig/memcached에서 CACHESIZE 변수는 기본적으로 64로 설정되며, 이는 64메가바이트를 의미합니다. 최대 캐시 크기보다 더 많은 데이터를 캐시에 삽입하면 캐시를 채우고 일부 항목은 memcached-tool을 사용하여 제거되며 memcached.service가 약 64MB의 메모리를 사용하고 있음을 알 수 있습니다.

```
[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 속성을 사용하여 memcached.service에 대한 하드 제한을 설정하며, 이 제한에 도달하면 OOM 킬러가 간접 호출됩니다. 재정의 파일에 추가하여 서비스에 대한 추가 옵션을 설정할 수 있습니다. 이 작업은 /etc/systemd/system/memcached.service.d/override.conf 파일을 직접 편집하거나 systemctl의 edit 명령을 사용하여 대화식으로 수행할 수 있습니다.

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

아래를 재정의에 추가하여 서비스에 대한 메모리 하드 제한을 32MB로 설정합니다.

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

6. systemd에 구성을 다시 로드하도록 지시

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

7. 이제 memcached.service가 32MB의 메모리 제한으로 실행 중인지 확인합니다.

```
[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. 서비스는 32MB 미만의 메모리를 사용하는 동안 정상적으로 작동하며, 32MB 미만의 임의 데이터를 캐시에 로드한 다음 서비스 상태를 확인하여 확인할 수 있습니다.

```
[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. 이제 기본 memcached 구성인 64MB의 전체 캐시를 사용하려고 시도하여 memcached가 32MB 이상의 메모리를 사용하도록 할 수 있습니다.

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

위 명령 중 어느 시점에 memcached 서버에 연결 오류가 있는 것을 확인할 수 있습니다. 이는 OOM Killer가 프로세스에 적용한 제한으로 인해 프로세스를 종료했기 때문입니다. 나머지 시스템은 정상적으로 작동하며 OOM Killer는 다른 프로세스를 고려하지 않습니다. 제한한 대상이 memcached.service뿐이기 때문입니다.

```
[ec2-user ~]$ sudo systemctl status memcached.service
# memcached.service - memcached daemon
   Loaded: loaded (/usr/lib/systemd/system/memcached.service; enabled; preset:
disabled)
   Drop-In: /etc/systemd/system/memcached.service.d
           ##override.conf
```

```
Active: failed (Result: oom-kill) since Fri 2025-01-31 23:20:28 UTC; 2s ago
Duration: 2.901s
Process: 360130 ExecStart=/usr/bin/memcached -p ${PORT} -u ${USER} -m
${CACHESIZE} -c ${MAXCONN} $OPTIONS (code=killed, signal=KILL)
Main PID: 360130 (code=killed, signal=KILL)
CPU: 94ms
```

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

cgroups를 사용하여 AL2023에서 프로세스 리소스 사용 제한

[systemd를 사용한 리소스 제어](#) 섹션을 사용하는 것이 좋지만, 이 섹션에서는 프로세스의 CPU 및 메모리 사용량을 제한하기 위한 기본 `libcgroup-tools` 유틸리티의 기본 사용량을 다룹니다. 두 방법 모두 이전에 [EPEL](#)에서 제공되었던, [cpulimit](#) 유틸리티를 사용하는 대체 방법입니다.

아래 예제에서는 `libcgroup-tools` 패키지의 유틸리티와 `sysfs`의 튜닝 가능 항목을 사용하여 CPU 및 메모리 사용량을 제한하면서 `stress-ng` 스트레스 테스트(`stress-ng` 패키지)를 실행하는 방법을 다룹니다.

명령줄에서 **`libcgroup-tools`**를 사용하여 리소스 사용 제한

1. `libcgroup-tools` 패키지를 설치합니다.

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

2. memory 및 cpu 컨트롤러를 사용하여 cgroup을 생성하고 이름(`our-example-limits`)을 지정합니다. `-a` 및 `-t` 옵션을 사용하여 `ec2-user` 사용자가 cgroup의 튜닝 가능 항목을 제어할 수 있도록 허용

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

이제 각 튜닝 가능 항목을 제어하는 데 사용할 수 있는 파일이 포함된 `/sys/fs/cgroup/our-example-limits/` 디렉터리가 있습니다.

Note

Amazon Linux 2는 AL2023에서 사용되는 `cgroup-v2` 대신 `cgroup-v1`을 사용합니다. AL2에서는 `sysfs` 경로가 다르며, `cgroup`의 제한을 제어하는 데 사용할 수 있는 파일이 포함된 `ec2-user` 소유의 `/sys/fs/cgroup/memory/our-example-limits` 및 `/sys/fs/cgroup/cpu/our-example-limits` 디렉터리가 있습니다.

3. `cgroup`의 모든 프로세스의 메모리 사용량을 1억 바이트로 제한합니다.

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

Note

Amazon Linux 2는 Amazon Linux 2023에서 사용하는 `cgroup-v2` 대신 `cgroup-v1`을 사용합니다. 즉, 일부 튜닝 가능 항목이 다릅니다. AL2의 메모리 사용량을 제한하기 위해 아래 튜닝 가능 항목이 대신 사용됩니다.

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

4. `cgroup`의 모든 프로세스의 CPU 사용량을 10%로 제한합니다. `cpu.max` 파일의 형식은 `$MAX $PERIOD`이며, 그룹을 모든 `$PERIOD`에 대해 `$MAX` 소비로 제한합니다.

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

Amazon Linux 2는 Amazon Linux 2023에서 사용하는 `cgroup-v2` 대신 `cgroup-v1`을 사용합니다. 즉, CPU 사용량을 제한하는 방법을 포함하여 일부 튜닝 가능 항목이 다릅니다.

5. 아래 예제는 `our-example-limits` `cgroup`에서 `stress-ng`(`dnf install -y stress-ng`를 실행하여 설치할 수 있음)를 실행합니다. `stress-ng` 명령이 실행되는 동안 `top`을 사용하여 CPU 시간의 10%로 제한되는 것을 관찰할 수 있습니다.

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

6. `cgroup`을 제거하여 정리

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

[cgroup-v2용 Linux 커널 설명서](#)에는 작동 방식에 대한 광범위한 세부 정보가 포함되어 있습니다. [cpu](#) 및 [메모리](#) 컨트롤러 설명서에서는 각 튜닝 가능 옵션을 사용하는 방법에 대한 세부 정보를 다룹니다.

에서 AL2023 사용 AWS

다른 AWS 서비스와 함께 사용하도록 AL2023을 설정할 수 있습니다. 예를 들어 [Amazon Elastic Compute Cloud](#)(Amazon EC2) 인스턴스를 실행할 때 AL2023 AMI를 선택할 수 있습니다.

이러한 설정 절차에서는 AWS Identity and Access Management (IAM) 서비스를 사용합니다. IAM에 대한 자세한 내용은 다음 참조 자료를 참조하세요.

- [AWS Identity and Access Management \(IAM\)](#)
- [IAM 사용 설명서](#)

주제

- [시작하기 AWS](#)
- [Amazon EC2 AL2023](#)
- [컨테이너에서 AL2023 사용](#)
- [AWS Elastic Beanstalk의 AL2023](#)
- [AWS CloudShell에서 AL2023 사용](#)
- [AL2023 기반 Amazon ECS AMI를 사용하여 컨테이너화된 워크로드 호스팅](#)
- [AL2023에서 Amazon Elastic File System 사용](#)
- [AL2023에서 Amazon EMR 사용](#)
- [에서 AL2023 사용 AWS Lambda](#)

시작하기 AWS

에 가입 AWS 계정

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. <https://portal.aws.amazon.com/billing/signup>을 엽니다.
2. 온라인 지시 사항을 따르세요.

등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

에 가입하면 AWS 계정 루트 사용자의 계정이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 [루트 사용자 액세스 권한이 필요한 작업](#)을 수행하는 것입니다.

AWS 는 가입 프로세스가 완료된 후 확인 이메일을 보냅니다. 언제든지 <https://aws.amazon.com/>으로 이동하고 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

에 가입한 후 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 AWS 계정보호 AWS IAM Identity Center, AWS 계정 루트 사용자 활성화 및 생성합니다.

보안 AWS 계정 루트 사용자

1. 루트 사용자를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 소유자 [AWS Management Console](#)로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자를 사용하여 로그인하는 데 도움이 필요하면 AWS Sign-In 사용 설명서의 [루트 사용자 로 로그인](#)을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

지침은 IAM 사용 설명서의 [AWS 계정 루트 사용자\(콘솔\)에 대한 가상 MFA 디바이스 활성화를 참조하세요](#).

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [AWS IAM Identity Center 설정](#)을 참조하세요.

2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

를 자격 증명 소스 IAM Identity Center 디렉터리로 사용하는 방법에 대한 자습서는 사용 AWS IAM Identity Center 설명서의 [기본값으로 사용자 액세스 구성을 IAM Identity Center 디렉터리 참조하세요](#).

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하다면 AWS Sign-In 사용 설명서의 [AWS 액세스 포털에 로그인](#)을 참조하세요.

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [Create a permission set](#)를 참조하세요.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

지침은 AWS IAM Identity Center 사용 설명서의 [그룹 추가](#)를 참조하세요.

프로그래밍 방식 액세스 권한 부여

사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 AWS Management Console. 프로그래밍 방식 액세스를 부여하는 방법에는 액세스하는 사용자 유형에 따라 다릅니다 AWS.

사용자에게 프로그래밍 방식 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장) 콘솔 자격 증명을 임시 자격 증명으로 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS 로컬 개발을 위한 로그인을 참조하세요. • AWS SDKs 경우 SDK 및 도구 참조 안내서의 AWS 로컬

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
		개발을 위한 로그인 을 참조하세요. AWS SDKs
작업 인력 ID (IAM Identity Center에서 관리되는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. - 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center 참조하세요. - AWS SDKs, 도구 및 AWS APIs 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWS SDKs
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용 의 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. - 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 IAM 사용자 자격 증명을 사용한 인증을 참조하세요. - AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWS SDKs - AWS APIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

Amazon EC2 AL2023

다음 절차 중 하나를 사용하여 AL2023 AMI로 Amazon EC2 인스턴스를 시작합니다. 표준 AMI 또는 미니멀 AMI를 선택합니다. 표준 AMI와 미니멀 AMI 차이에 대한 자세한 내용은 [AL2023 표준\(기본\) AMI와 미니멀 AMI 비교](#)을 참조하세요.

주제

- [Amazon EC2 콘솔에서 AL2023 시작](#)
- [SSM 파라미터 뭉치를 사용하여 AL2023 시작 AWS CLI](#)
- [를 사용하여 최신 AL2023 AMI 시작 CloudFormation](#)
- [특정 AMI ID로 AL2023 실행](#)
- [AL2023 AMI 사용 중단 및 수명 주기](#)
- [AL2023 인스턴스에 연결](#)
- [AL2023 표준 AMI와 최소 AMI 비교](#)

Amazon EC2 콘솔에서 AL2023 시작

Amazon EC2 콘솔에서 AL2023 AMI를 시작합니다.

Note

ARM 기반 인스턴스에서 AL2023은 Graviton2 이상 프로세서를 사용하는 인스턴스 유형만 지원합니다. AL2023은 A1 인스턴스를 지원하지 않습니다.

다음 단계를 거쳐 Amazon EC2 콘솔의 AL2023 AMI로 Amazon EC2 인스턴스를 시작합니다.

AL2023 AMI로 EC2 인스턴스를 시작하는 방법

1. <https://console.aws.amazon.com/ec2/>에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 AMI를 선택합니다.
3. 드롭다운 메뉴에서 공개 이미지(Public images)를 선택합니다.
4. 검색 필드에 **al2023-ami**를 입력합니다.

Note

소유자 별칭에 amazon이 표시되는지 확인하세요.

5. 목록에서 이미지를 선택합니다. 소스에서 AMI가 표준인지 미니멀인지 확인할 수 있습니다. AL2023 AMI 이름은 다음 형식으로 해석할 수 있습니다.

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

6. 다음 이미지는 AL2023 AMI 일부 목록입니다.

Amazon Machine Images (AMIs) (4) Info						
Public images al2023-ami Search Clear filters						
☐	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 인스턴스를 시작하는 방법에 대한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 Linux 인스턴스 시작](#) 섹션을 참조하세요.

SSM 파라미터 값을 사용하여 AL2023 시작 AWS CLI

에서 AMI의 SSM 파라미터 값을 사용하여 AL2023의 새 인스턴스를 시작할 AWS CLI 수 있습니다. 구체적으로 설명하면, 다음 목록의 동적 SSM 파라미터 값 중 하나를 SSM 파라미터 `/aws/service/ami-amazon-linux-latest/` 값 앞에 넣으세요. 이렇게 하면 AWS CLI 인스턴스를 시작할 수 있습니다.

- arm64 아키텍처 `al2023-ami-kernel-default-arm64`
- arm64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-default-arm64`
- x86_64 아키텍처 `al2023-ami-kernel-default-x86_64`
- x86_64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-default-x86_64`

Note

항목은 예제 파라미터입니다. 이를 사용자의 정보로 대체합니다.

```
$ 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` 플래그가 SSM 파라미터의 값을 지정합니다.

`--instance-type` 플래그는 인스턴스의 유형과 크기를 지정합니다. 이 플래그는 선택한 AMI 유형과 호환되어야 합니다.

`--region` 플래그는 인스턴스를 생성하는 AWS 리전 를 지정합니다.

`--key-name` 플래그는 인스턴스에 연결하는 데 사용되는 AWS 리전 키를 지정합니다. 인스턴스를 만든 리전에 있는 키를 사용하지 않으면 SSH를 사용하여 인스턴스에 연결할 수 없습니다.

`--security-group-ids` 플래그는 인바운드 및 아웃바운드 네트워크 트래픽에 대한 액세스 권한을 결정하는 보안 그룹을 지정합니다.

⚠ Important

에서는 포트를 통해 원격 시스템에서 인스턴스에 액세스할 수 있는 기존 보안 그룹을 지정 AWS CLI 해야 합니다 TCP:22. 지정된 보안 그룹이 없으면 새 인스턴스는 기본 보안 그룹에 배치됩니다. 기본 보안 그룹의 인스턴스는 VPC 내의 다른 인스턴스만 연결할 수 있습니다.

자세한 내용을 알아보려면 AWS Command Line Interface 사용 설명서의 [Amazon EC2 인스턴스 시작, 목록 작성 및 종료](#)를 참조하세요.

를 사용하여 최신 AL2023 AMI 시작 CloudFormation

를 사용하여 AL2023 AMI를 시작하려면 다음 템플릿 중 하나를 CloudFormation 사용합니다.

i Note

x86_64 및 Arm64 AMI는 각각 다른 인스턴스 유형이 필요합니다. 자세한 내용은 [Amazon EC2 인스턴스 유형](#)을 참조하세요.

JSON 템플릿:

```
{
  "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 템플릿:

```
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
```

필요한 경우 “Default” 섹션 마지막에 있는 AMI 파라미터를 교체하세요. 다음과 같은 파라미터 값을 사용할 수 있습니다.

- arm64 아키텍처 al2023-ami-kernel-6.1-arm64
- arm64 아키텍처 (미니멀 AMI) al2023-ami-minimal-kernel-6.1-arm64
- x86_64 아키텍처 al2023-ami-kernel-6.1-x86_64
- x86_64 아키텍처 (미니멀 AMI) al2023-ami-minimal-kernel-6.1-x86_64

다음은 동적 커널 사양입니다. 기본 커널 버전은 주요 커널 버전이 업데이트될 때마다 자동으로 변경됩니다.

- arm64 아키텍처 al2023-ami-kernel-default-arm64
- arm64 아키텍처 (미니멀 AMI) al2023-ami-minimal-kernel-default-arm64
- x86_64 아키텍처 al2023-ami-kernel-default-x86_64
- x86_64 아키텍처 (미니멀 AMI) al2023-ami-minimal-kernel-default-x86_64

특정 AMI ID로 AL2023 실행

AMI ID를 사용하여 특정 AL2023 AMI를 시작할 수 있습니다. Amazon EC2 콘솔의 AMI 목록을 보고 어떤 AL2023 AMI ID가 필요한지 확인할 수 있습니다. 또는를 사용할 수 있습니다 AWS Systems

Manager. Systems Manager를 사용하는 경우 이전 섹션에 나온 AMI 별칭 중에서 선택하세요. 자세한 내용은 [AWS Systems Manager Parameter Store를 사용하여 최신 Amazon Linux AMI IDs](#).

AL2023 AMI 사용 중단 및 수명 주기

새 AL2023 릴리스에는 새 AMI가 포함됩니다. AMI를 등록하면 지원 중단 날짜가 표시됩니다. AL2023 AMI 지원 중단일은 각 개별 커널 릴리스의 [AL2023의 커널 라이브 패치](#) 기간과 일치하는 출시된 시점으로부터 90일입니다.

Note

90일 개별 AMI 지원 종단을 의미하며 AL2023 [릴리스 케이던스](#) 또는 제품 지원 기간을 의미하지는 않습니다.

AMI의 지원 종단에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 [AMI 지원 중단](#) 섹션을 참조하세요.

업데이트된 AMI를 정기적으로 사용하여 인스턴스를 시작하면 업데이트된 커널을 포함한 최신 보안으로 업데이트된 인스턴스를 시작할 수 있습니다. 구 AMI 버전을 실행하여 업데이트한다면 인스턴스에 최신 보안 업데이트가 적용되지 않는 기간이 있습니다. 최신 AMI를 사용 중인지 확인하려면 SSM 파라미터를 사용하는 것이 좋습니다.

SSM 파라미터를 사용하여 인스턴스를 시작하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- [SSM 파라미터 값을 사용하여 AL2023 시작 AWS CLI](#)
- [를 사용하여 최신 AL2023 AMI 시작 CloudFormation](#)

AL2023 인스턴스에 연결

SSH 또는 AWS Systems Manager 를 사용하여 AL2023 인스턴스에 연결합니다.

SSH를 사용하여 인스턴스에 연결

SSH를 사용하여 인스턴스에 연결하는 방법에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [SSH를 사용하여 Linux 인스턴스에 연결](#) 섹션을 참조하세요.

를 사용하여 인스턴스에 연결 AWS Systems Manager

를 사용하여 AL2023 인스턴스에 AWS Systems Manager 연결하는 방법에 대한 지침은 Amazon EC2 사용 설명서의 [세션 관리자를 사용하여 Linux 인스턴스에 연결을 참조하세요](#).

Amazon EC2 Instance Connect 사용

AL2023 AMI(최소 AMI 제외)에는 EC2 인스턴스 연결 에이전트가 기본적으로 설치되어 있습니다. 최소 AMI에서 실행된 AL2023 인스턴스 EC2 인스턴스 연결을 사용하려면 `ec2-instance-connect` 패키지를 설치해야 합니다. EC2 인스턴스 연결 사용에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [EC2 인스턴스 연결을 사용하여 Linux 인스턴스에 연결](#) 섹션을 참조하세요.

AL2023 표준 AMI와 최소 AMI 비교

표준(기본값) 또는 최소 AL2023 AMI를 사용하여 Amazon EC2 인스턴스를 시작할 수 있습니다. 표준 또는 최소 AMI 유형의 Amazon EC2 인스턴스를 시작하는 방법에 대한 지침은 [Amazon EC2 AL2023](#) 섹션을 참조하세요.

표준 AL2023 AMI는 가장 일반적으로 사용되는 애플리케이션 및 도구와 함께 설치됩니다. 빠르게 시작하고 싶어서 AMI 사용자 지정을 사용하지 않는다면 표준 AMI를 사용하는 것이 좋습니다.

최소 AL2023 AMI는 운영 체제(OS)를 실행하는 데 필요한 가장 기본적인 도구와 유틸리티만 포함한 약식 기본 버전입니다. OS 설치 공간을 최대한 줄이려면 미니멀 AMI를 사용하는 것이 좋습니다. 미니멀 AMI로 디스크 공간 사용률을 조금 줄이고 장기적인 비용 효율성을 향상시킬 수 있습니다. 크기가 작은 OS를 사용하고 도구와 애플리케이션을 수동으로 설치해도 좋다면 미니멀 AMI가 적합합니다.

컨테이너 이미지는 AL2023 미니멀 AMI 패키지 세트에 더 가깝습니다.

Amazon Linux 2023 Images에 설치된 패키지 비교

AL2023 AMI, 최소 AMI 및 컨테이너 이미지에 있는 RPM 간의 비교입니다.

패키지	AMI	미니멀 AMI	컨테이너
<code>acl</code>	2.3.1		
<code>acpid</code>	2.0.32		
<code>alternatives</code>	1.15	1.15	1.15
<code>amazon-chrony-config</code>	4.3	4.3	

패키지	AMI	미니멀 AMI	컨테이너
amazon-ec2-net-utils	2.5.1	2.5.1	
amazon-linux-repo-cdn			2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1	
amazon-rpm-config	228		
amazon-ssm-agent	3.3.987.0		
amd-ucode-firmware	20210208(noarch)	20210208(noarch)	
at	3.1.23		
attr	2.5.1		
audit	3.0.6	3.0.6	
audit-libs	3.0.6	3.0.6	3.0.6
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

패키지	AMI	미니멀 AMI	컨테이너
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
ca-certificates	2023년 2월 68일	2023년 2월 68일	2023년 2월 68일
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	

패키지	AMI	미니멀 AMI	컨테이너
cloud-utils-growpart	0.31	0.31	
coreutils	8.32	8.32	
coreutils-common	8.32	8.32	
coreutils-single			8.32
cpio	2.13	2.13	
cracklib	2.9.6	2.9.6	
cracklib-dicts	2.9.6	2.9.6	
crontabs	1.11		
crypto-policies	20220428	20220428	20220428
crypto-policies-scripts	20220428		
cryptsetup	2.6.1		
cryptsetup-libs	2.6.1	2.6.1	
curl-minimal	8.5.0	8.5.0	8.5.0
cyrus-sasl-lib	2.1.27	2.1.27	
cyrus-sasl-plain	2.1.27		
dbus	1.12.28	1.12.28	
dbus-broker	32	32	

패키지	AMI	미니멀 AMI	컨테이너
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-n-release-notification	1.2	1.2	
dnf-plugins-core	4.3.0	4.3.0	
dnf-plugin-support-info	1.2	1.2	
dnf-utils	4.3.0		
dosfstools	4.2		
dracut	055	055	
dracut-config-ec2	3.0	3.0	
dracut-config-generic	055	055	
dwz	0.14		

패키지	AMI	미니멀 AMI	컨테이너
dyninst	10.2.1		
e2fsprogs	1.46.5	1.46.5	
e2fsprogs-libs	1.46.5	1.46.5	
ec2-hibinit-agent	1.0.8		
ec2-instance-connect	1.1		
ec2-instance-connect-selinux	1.1		
ec2-utils	2.2.0	2.2.0	
ed	1.14.2		
efi-filesystem	5	5	
efi-srpm-macros	5		
efivar	38	38	
efivar-libs	38	38	
elfutils-debuginfod-client	0.188		
elfutils-default-yama-scope	0.188	0.188	0.188
elfutils-libelf	0.188	0.188	0.188

패키지	AMI	미니멀 AMI	컨테이너
elfutils-libs	0.188	0.188	0.188
ethtool	5.15		
expat	2.5.0	2.5.0	2.5.0
file	5.39	5.39	
file-libs	5.39	5.39	5.39
filesystem	3.14	3.14	3.14
findutils	4.8.0	4.8.0	
fonts-srpm-macros	2.0.5		
fstrm	0.6.1		
fuse-libs	2.9.9	2.9.9	
gawk	5.1.0	5.1.0	5.1.0
gdbm-libs	1.19	1.19	1.19
gdisk	1.0.8	1.0.8	
gettext	0.21	0.21	
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	

패키지	AMI	미니멀 AMI	컨테이너
glibc-common	2.34	2.34	2.34
glibc-gconv-extra	2.34		
glibc-locale-source	2.34	2.34	
glibc-minimal-langpack			2.34
gmp	6.2.1	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7	2.3.7
gnutls	3.8.0	3.8.0	
go-srpm-macros	3.2.0		
gpgme	1.15.1	1.15.1	1.15.1
gpm-libs	1.20.7		
grep	3.8	3.8	3.8
groff-base	1.22.4	1.22.4	
grub2-common	2.06	2.06	
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	

패키지	AMI	미니멀 AMI	컨테이너
grub2-tools-minimal	2.06	2.06	
grubby	8.40	8.40	
gssproxy	0.8.4		
gzip	1.12	1.12	
hostname	3.23	3.23	
hunspell	1.7.0		
hunspell-en	0.20140811.1		
hunspell-en-GB	0.20140811.1		
hunspell-en-US	0.20140811.1		
hunspell-filesystem	1.7.0		
hwdata	0.384	0.384	
info	6.7		
inih	49	49	
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		

패키지	AMI	미니멀 AMI	컨테이너
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-libvepatch-repos3	2023.6.20241031	2023.6.20241031	
kernel-srpm-macros	1.0		
kernel-tools	6.1.112		
keyutils	1.6.3		
keyutils-libs	1.6.3	1.6.3	1.6.3
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

패키지	AMI	미니멀 AMI	컨테이너
libaio	0.3.111		
libarchive	3.7.4	3.7.4	3.7.4
libargon2	20171227	20171227	
libassuan	2.5.5	2.5.5	2.5.5
libattr	2.5.1	2.5.1	2.5.1
libbasicobjects	0.1.1		
libblkid	2.37.4	2.37.4	2.37.4
libcap	2.48	2.48	2.48
libcap-ng	0.8.2	0.8.2	0.8.2
libcbor	0.7.0	0.7.0	
libcollection	0.7.0		
libcom_err	1.46.5	1.46.5	1.46.5
libcomps	0.1.20	0.1.20	0.1.20
libconfig	1.7.2		
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	

패키지	AMI	미니멀 AMI	컨테이너
libev	4.33		
libevent	2.1.12		
libfdisk	2.37.4	2.37.4	
libffi	3.4.4	3.4.4	3.4.4
libfido2	1.10.0	1.10.0	
libgcc	11.4.1	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2	1.10.2
libgomp	11.4.1	11.4.1	11.4.1
libgpg-error	1.42	1.42	1.42
libibverbs	48.0		
libidn2	2.3.2	2.3.2	2.3.2
libini_config	1.3.1		
libkcapi	1.4.0	1.4.0	
libkcapi-hmacalc	1.4.0	1.4.0	
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

패키지	AMI	미니멀 AMI	컨테이너
libnfsidmap	2.5.4		
libnhttp2	1.59.0	1.59.0	1.59.0
libnl3	3.5.0		
libpath_utils	0.2.1		
libpcap	1.10.1		
libpipeline	1.5.3	1.5.3	
libpkgconf	1.8.0		
libpsl	0.21.1	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4	
libref_array	0.1.5		
librepo	1.14.5	1.14.5	1.14.5
libreport-filesystem	2.15.2	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3	
libselinux	3.4	3.4	3.4
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

패키지	AMI	미니멀 AMI	컨테이너
libsolv	0.7.22	0.7.22	0.7.22
libss	1.46.5	1.46.5	
libsss_certmap	2.9.4		
libsss_idmap	2.9.4		
libsss_nss_idmap	2.9.4		
libsss_sudo	2.9.4		
libstdc++	11.4.1	11.4.1	11.4.1
libstoragegmt	1.9.4		
libtalloc	2.3.4		
libtasn1	4.19.0	4.19.0	4.19.0
libtdb	1.4.7		
libtevent	0.13.0		
libtextstyle	0.21	0.21	
libtirpc	1.3.3		
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

패키지	AMI	미니멀 AMI	컨테이너
libverto-libev	0.3.2		
libxcrypt	4.4.33	4.4.33	4.4.33
libxml2	2.10.4	2.10.4	2.10.4
libyaml	0.2.5	0.2.5	0.2.5
libzstd	1.5.5	1.5.5	1.5.5
linux-firmware-whence	20210208(noarch)	20210208(noarch)	
lm_sensors-libs	3.6.0		
lmdb-libs	0.9.29		
logrotate	3.20.1	3.20.1	
lsof	4.94.0		
lua-libs	5.4.4	5.4.4	5.4.4
lua-srpm-macros	1		
lz4-libs	1.9.4	1.9.4	1.9.4
man-db	2.9.3	2.9.3	
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

패키지	AMI	미니멀 AMI	컨테이너
ncurses-libs	6.2	6.2	6.2
nettle	3.8	3.8	
net-tools	2.0	2.0	
newt	0.52.21		
nfs-utils	2.5.4		
npth	1.6	1.6	1.6
nspr	4.35.0		
nss	3.90.0		
nss-softokn	3.90.0		
nss-softokn-freebl	3.90.0		
nss-sysinit	3.90.0		
nss-util	3.90.0		
ntsysv	1.15		
numactl-libs	2.0.14	2.0.14	
ocaml-srpm-macros	6		
oniguruma	6.9.7.1	6.9.7.1	
openblas-srpm-macros	2		
openldap	2.4.57	2.4.57	

패키지	AMI	미니멀 AMI	컨테이너
openssh	8.7p1	8.7p1	
openssh-clients	8.7p1	8.7p1	
openssh-server	8.7p1	8.7p1	
openssl	3.0.8	3.0.8	
openssl-lib	3.0.8	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12	
os-prober	1.77	1.77	
p11-kit	0.24.1	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1	0.24.1
package-notes-srpm-macros	0.4		
pam	1.5.1	1.5.1	
parted	3.4		
passwd	0.80	0.80	
pciutils	3.7.0	3.7.0	
pciutils-lib	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		

패키지	AMI	미니멀 AMI	컨테이너
perl-constant	1.33		
perl-DynaLoader	1.47		
perl-Encode	3.15		
perl-Errno	1.30		
perl-Exporter	5.74		
perl-Fcntl	1.13		
perl-File-Basename	2.85		
perl-File-Path	2.18		
perl-File-stat	1.09		
perl-File-Temp	0.231.100		
perl-Getopt-Long	2.52		
perl-Getopt-Std	1.12		
perl-HTTP-Tiny	0.078		
perl-if	0.60.800		
perl-interpreter	5.32.1		
perl-IO	1.43		
perl-IPC-Open3	1.21		
perl-libs	5.32.1		

패키지	AMI	미니멀 AMI	컨테이너
perl-MIME-Base64	3.16		
perl-mro	1.23		
perl-overload	1.31		
perl-overloading	0.02		
perl-parent	0.238		
perl-PathTools	3.78		
perl-Pod-Escapes	1.07		
perl-podlators	4.14		
perl-Pod-Perldoc	3.28.01		
perl-Pod-Simple	3.42		
perl-Pod-Usage	2.01		
perl-POSIX	1.94		
perl-Scalar-List-Utils	1.56		
perl-SelectSaver	1.02		
perl-Socket	2.032		
perl-srpm-macros	1		

패키지	AMI	미니멀 AMI	컨테이너
perl-Storable	3.21		
perl-subs	1.03		
perl-Symbol	1.08		
perl-Term-ANSIColor	5.01		
perl-Term-Cap	1.17		
perl-Text-ParseWords	3.30		
perl-Text-Tabs+Wrap	2021.0726		
perl-Time-Local	1.300		
perl-vars	1.05		
pkgconf	1.8.0		
pkgconf-m4	1.8.0		
pkgconf-pkg-config	1.8.0		
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		

패키지	AMI	미니멀 AMI	컨테이너
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	2019년 0월 19일	2019년 0월 19일	
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	

패키지	AMI	미니멀 AMI	컨테이너
python3-dnf	4.14.0	4.14.0	4.14.0
python3-dnf-plugins-core	4.3.0	4.3.0	
python3-docutils	0.16	0.16	
python3-gpg	1.15.1	1.15.1	1.15.1
python3-hawkey	0.69.0	0.69.0	0.69.0
python3-idna	2.10	2.10	
python3-jinja2	2.11.3	2.11.3	
python3-jmespath	0.10.0	0.10.0	
python3-jsonpatch	1.21	1.21	
python3-jsonpointer	2.0	2.0	
python3-jsonschema	3.2.0	3.2.0	
python3-libibcomps	0.1.20	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16	3.9.16
python3-libselinux	3.4	3.4	

패키지	AMI	미니멀 AMI	컨테이너
python3-l ibsemanage	3.4	3.4	
python3-l ibstoragemgmt	1.9.4		
python3-l ockfile	0.12.2		
python3-m arkupsafe	1.1.1	1.1.1	
python3-n etifaces	0.10.6	0.10.6	
python3-o authlib	3.0.2	3.0.2	
python3-pip- wheel	21.3.1	21.3.1	21.3.1
python3-ply	3.11	3.11	
python3-p olicycoreutils	3.4	3.4	
python3-p rettytable	0.7.2	0.7.2	
python3-prompt- toolkit	3.0.24	3.0.24	
python3-p ycparser	2.20	2.20	
python3-p yrsistent	0.17.3	0.17.3	

패키지	AMI	미니멀 AMI	컨테이너
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	
python3-setuptools	59.6.0	59.6.0	
python3-setuptools-wheel	59.6.0	59.6.0	59.6.0
python3-six	1.15.0	1.15.0	
python3-systemd	235	235	
python3-urllib3	1.25.10	1.25.10	
python3-wcwidth	0.2.5	0.2.5	
python-chevron	0.13.1		

패키지	AMI	미니멀 AMI	컨테이너
python-srpm-macros	3.9		
quota	4.06		
quota-nls	4.06		
readline	8.1	8.1	8.1
rng-tools	6.14	6.14	
rootfiles	8.1	8.1	
rpcbind	1.2.6		
rpm	4.16.1.3	4.16.1.3	4.16.1.3
rpm-build-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-libs	4.16.1.3	4.16.1.3	4.16.1.3
rpm-plugin-selinux	4.16.1.3	4.16.1.3	
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

패키지	AMI	미니멀 AMI	컨테이너
selinux-policy	38.1.45	38.1.45	
selinux-policy-targeted	38.1.45	38.1.45	
setup	2.13.7	2.13.7	2.13.7
shadow-utils	4.9	4.9	
slang	2.3.2		
sqlite-libs	3.40.0	3.40.0	3.40.0
sssd-client	2.9.4		
sssd-common	2.9.4		
sssd-kcm	2.9.4		
sssd-nfs-idmap	2.9.4		
strace	6.8		
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	

패키지	AMI	미니멀 AMI	컨테이너
systemd-resolved	252.23	252.23	
systemd-udev	252.23	252.23	
system-release	2023.6.20241031	2023.6.20241031	2023.6.20241031
systemtap-runtime	4.8		
tar	1.34	1.34	
tbb	2020.3		
tcpdump	4.99.1		
tcsh	6.24.07		
time	1.9		
traceroute	2.1.3		
tzdata	2024a	2024a	2024a
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		

패키지	AMI	미니멀 AMI	컨테이너
vim-filesystem	9.0.2153		
vim-minimal	9.0.2153	9.0.2153	
wget	1.21.3		
which	2.21	2.21	
words	3.0		
xfsdump	3.1.11		
xfsprogs	5.18.0	5.18.0	
xxd	9.0.2153		
xxhash-libs	0.8.0		
xz	5.2.5	5.2.5	
xz-libs	5.2.5	5.2.5	5.2.5
yum	4.14.0	4.14.0	4.14.0
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	

컨테이너에서 AL2023 사용

Note

Amazon ECS에서 AL2023으로 컨테이너 워크로드를 호스팅하는 방법에 대한 자세한 내용은 [AL2023 Amazon ECS 컨테이너 호스트](#) 섹션을 참조하세요.

사용 사례에 따라 컨테이너 내에서 AL2023을 사용할 수 있는 몇 가지 방법이 있습니다. [AL2023 기본 컨테이너 이미지](#)는 Amazon Linux 2 컨테이너 이미지와 AL2023 최소 AMI와 가장 유사합니다.

고급 사용자에게는 [기본 컨테이너](#)를 빌드하는 방법을 설명하는 설명서와 함께 AL2023.2 릴리스에 도입된 최소 컨테이너 이미지를 제공합니다.

AL2023에서도 AL2023 기반 컨테이너 이미지 또는 Linux 배포판 기반 컨테이너 등 컨테이너 워크로드를 호스팅할 수 있습니다. [AL2023 Amazon ECS 컨테이너 호스트](#) 또는 설치된 컨테이너 런타임 패키지를 사용할 수 있습니다. docker, containerd, 및 nerdctl 패키지를 AL2023에 설치하고 사용할 수 있습니다.

주제

- [AL2023 기본 컨테이너 이미지 사용](#)
- [AL2023 최소 컨테이너 이미지](#)
- [기본 AL2023 컨테이너 이미지 빌드](#)
- [Amazon Linux 2023 컨테이너 이미지에 설치된 패키지 비교](#)
- [Amazon Linux 2023 미니멀 AMI와 컨테이너 이미지에 설치된 패키지 비교](#)

AL2023 기본 컨테이너 이미지 사용

AL2023 컨테이너 이미지는 AL2023 AMI에 포함된 동일한 소프트웨어 구성 요소로부터 빌드됩니다. 이는 Docker 워크로드의 기본 이미지로 어느 환경에나 사용할 수 있습니다. [Amazon Elastic Compute Cloud](#) Amazon EC2) 애플리케이션에 Amazon Linux AMI를 이미 사용하고 있는 경우, Amazon Linux 컨테이너 이미지를 사용하여 애플리케이션을 컨테이너화할 수 있습니다.

로컬 개발 환경에서 Amazon Linux 컨테이너 이미지를 사용한 다음 Amazon Elastic Container Service(Amazon ECS)를 AWS 사용하여 애플리케이션을 로 푸시합니다. <https://docs.aws.amazon.com/AmazonECS/latest/userguide/> 자세한 정보는 Amazon Elastic Container Registry 사용 설명서의 [Amazon ECS에서 Amazon ECR 이미지 사용](#)을 참조하세요.

Amazon Linux 컨테이너 이미지는 Amazon ECR Public에서 사용할 수 있습니다. 지정된 AWS 대리인을 통해 또는 GitHub의 [amazon-linux-2023](https://github.com/amazonlinux/amazon-linux-2023) 리포지토리에 문제를 제출하여 AL2023에 대한 피드백을 제공할 수 있습니다. <https://github.com/amazonlinux/amazon-linux-2023/issues>

Amazon ECR Public에서 Amazon Linux 컨테이너 이미지를 가져오려면

1. Docker 클라이언트를 Amazon Linux Public 레지스트리에 인증합니다. 인증 토큰은 12시간 동안 유효합니다. 자세한 내용은 Amazon Elastic Container Registry 사용 설명서의 [프라이빗 레지스트리 권한](#)을 참조하세요.

Note

get-login-password 명령은 최신 버전의 AWS CLI 버전 2를 사용하여 지원됩니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 [AWS Command Line Interface 설치](#)를 참조하세요.

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

출력값은 다음과 같습니다.

```
Login succeeded
```

2. docker pull 명령을 실행하여 Amazon Linux 컨테이너 이미지를 가져옵니다. Amazon ECR Public Gallery에서 Amazon Linux 컨테이너 이미지를 보려면 [Amazon ECR Public Gallery - amazonlinux](#)를 참조하세요.

Note

AL2023 Docker 컨테이너 이미지를 가져올 때 다음 형식 중 하나로 태그를 사용할 수 있습니다.

- 최신 AL2023 컨테이너 이미지 버전을 다운로드하려면 :2023 태그를 사용하세요.
- 다음 형식으로 AL2023 특정 버전을 다운로드하세요.
 - :2023.[0-7 release quarter].[release date].[build number]

다음은 :2023 태그를 사용하여 가장 최근의 AL2023 컨테이너 이미지를 가져오는 예시입니다.

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

3. (선택 사항) 로컬에서 컨테이너를 실행합니다.

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

Docker Hub에서 AL2023 컨테이너 이미지를 가져오려면

1. docker pull 명령을 사용하여 AL2023 컨테이너 이미지를 가져옵니다.

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

2. (선택 사항) 로컬에서 컨테이너를 실행합니다.

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

Note

AL2023 컨테이너 이미지는 dnf 패키지 관리자만 소프트웨어 패키지를 설치할 수 있습니다. 즉, 다른 소프트웨어에 사용할 수 있는 amazon-linux-extras 명령이나 이와 비슷한 명령이 없습니다.

AL2023 최소 컨테이너 이미지

Note

표준 AL2023 컨테이너 이미지는 대부분의 사용 사례에 적합하며, 최소 컨테이너 이미지에 맞추는 것이 AL2023 기본 컨테이너 이미지에 맞추는 것보다 번거롭습니다.

AL2023.2에 도입된 AL2023 최소 컨테이너 이미지는 다른 패키지를 설치하는 데 필요한 최소 패키지만 포함한다는 점에서 기본 컨테이너 이미지와 다릅니다. 최소 컨테이너 이미지는 편리한 패키지 세트가 아닌 최소 패키지 세트로 설계되었습니다.

AL2023 미니멀 컨테이너 이미지는 AL2023에 이미 설치된 소프트웨어 구성 요소를 기반으로 구축되었습니다. 최소 컨테이너 이미지의 주요 차이점은 microdnf를 사용하여 완전한 기능을 갖춘 Python 기반의 dnf가 아닌 dnf 패키지 관리자를 제공하는 것입니다. 이를 통해 AL2023 AMI 및 기본 컨테이너 이미지에 포함된 dnf 패키지 관리자의 전체 기능 세트가 없다는 단점이 있지만, 최소 컨테이너 이미지를 더 작게 만들 수 있습니다.

AL2023 최소 컨테이너 이미지는 `provided.al2023` AWS Lambda 런타임 환경의 기반을 형성합니다.

최소 컨테이너 이미지에 포함된 패키지의 자세한 목록은 [Amazon Linux 2023 컨테이너 이미지에 설치된 패키지 비교](#) 섹션을 참조하세요.

미니멀 컨테이너 이미지 사이즈

AL2023 최소 컨테이너 이미지는 AL2023 기본 컨테이너 이미지보다 패키지 수가 적기 때문에 크기도 훨씬 작습니다. 아래 표는 Amazon Linux의 현재 및 이전 릴리스의 컨테이너 이미지 옵션을 비교한 것입니다.

Note

이미지 크기는 [Amazon Linux Amazon ECR 퍼블릭 갤러리](#)에 있는 것과 같습니다.

이미지	버전	이미지 크기	Note
Amazon Linux 1(AL1)	2018년 3월 0일 2023년 9월 18일	62.3MB	x86-64 전용
Amazon Linux 2	2.0.20230926.0	64.2MB	aaarch64은 x86-64보다 1.6MB 큼니다.
Amazon Linux 기본 컨테이너 이미지	2023년 2월 23일 1002.0	52.4MB	

이미지	버전	이미지 크기	Note
Amazon Linux 2023 미니멀 컨테이너 이미지	2023.2.20231002.0-미니멀	35.2MB	

AL2023 미니멀 컨테이너 이미지 사용

ECR에서 AL2023 최소 컨테이너 이미지를 사용할 수 있으며 2023-minimal 태그는 항상 최신 AL2023 기반 최소 컨테이너 이미지를 의미하지만, minimal 태그는 AL2023 이후에 출시된 최신 Amazon Linux 버전으로 업데이트될 수 있습니다.

다음 예제와 함께 docker를 사용하여 이러한 태그를 가져올 수 있습니다.

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

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

다음은 최소 컨테이너 이미지를 가져와 GCC를 설치하는 Dockerfile의 예제입니다.

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

기본 AL2023 컨테이너 이미지 빌드

AL2023 컨테이너 이미지는 AL2023 AMI에 포함된 동일한 소프트웨어 구성 요소로부터 빌드됩니다. 여기에는 기본 컨테이너 계층이 Amazon EC2 인스턴스(예: 패키지 관리자 dnf)에서 실행되는 것과 유사하게 작동할 수 있게 하는 소프트웨어가 포함되어 있습니다. 이 섹션에서는 애플리케이션에 대한 최소한의 종속성만 포함하는 컨테이너를 처음부터 구성하는 방법을 설명합니다.

Note

표준 AL2023 컨테이너 이미지는 대부분의 사용 사례에 적합합니다. 표준 컨테이너 이미지를 사용하면 이미지 위에 쉽게 빌드할 수 있습니다. 기본 컨테이너 이미지를 사용하면 이미지 위에 빌드하기 더 어렵습니다.

애플리케이션에 대한 종속성을 최소화한 컨테이너를 만들려면

1. 런타임 종속 항목을 확인하세요. 이는 애플리케이션에 따라 다릅니다.
2. FROM scratch을 빌드하는 Dockerfile / Containerfile를 구성하세요. 다음 Dockerfile 예제를 사용하여 bash 셸과 그 종속성만 포함하는 컨테이너를 빌드할 수 있습니다.

```
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"]
```

- Dockerfile은 다음과 같은 방법으로 실행합니다.

1. build라는 이름의 AL2023 컨테이너를 실행합니다. 이 컨테이너는 기본 컨테이너를 부트스트랩하는 데 사용되며, 자체 배포되지 않고 배포할 컨테이너를 생성합니다.
2. /sysroot 디렉터리 생성 이 디렉터리에서 build 컨테이너가 기본 컨테이너에 필요한 종속성을 설치합니다. 다음 단계는 /sysroot 경로가 기본 이미지의 루트 디렉터리가 되도록 패키징됩니다.

이 방식으로 --installroot 옵션을 dnf로 사용하면 다른 AL2023 이미지가 만들어집니다. dnf 기능으로 설치 프로그램과 이미지 생성 도구를 사용할 수 있습니다.

3. dnf를 호출하여 /sysroot에 패키지를 설치합니다.

이 rpm -q system-release --qf '%{VERSION}' 명령은 system-release 패키지를 쿼리(-q)하고 쿼리 형식(--qf)을 설정하여 쿼리되는 패키지 버전(%{VERSION} 변수는 RPM 버전의 rpm 변수)을 출력합니다.

build 컨테이너에서 system-release 버전의 --releasever 인수를 dnf로 설정하면 업데이트된 Amazon Linux 컨테이너 기본 이미지가 릴리스될 때마다 Dockerfile을 사용하여 기본 컨테이너를 다시 빌드할 수 있습니다.

를 2023.10.20260325과 같은 `--releasever` Amazon Linux 2023 버전으로 설정할 수 있습니다. 이렇게 하면 `build` 컨테이너가 최신 AL2023 버전으로 실행되지만 현재 AL2023 릴리스와 관계없이 2023.10.20260325에서 베어본 컨테이너를 빌드합니다.

`--setopt=install_weak_deps=False` 구성 옵션으로 권장하거나 권장되는 종속성이 아닌 필수 종속성만 `dnf`에 설치할 수 있습니다.

4. 설치된 시스템을 빈 (FROM scratch) 컨테이너의 루트에 복사합니다.
5. `/bin/bash` 경우 `ENTRYPOINT`를 원하는 바이너리로 설정합니다.
3. 빈 디렉터리를 만들고 2단계의 예제 내용을 `Dockerfile` 파일에 추가합니다.

```
$ 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. 다음 명령을 실행하여 컨테이너를 빌드합니다.

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

5. 다음 명령으로 컨테이너를 실행하면 `bash` 전용 컨테이너가 얼마나 작은지 알 수 있습니다.

```
$ 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
```

좀 더 실용적인 예를 들면, 다음 방법으로 Hello World!를 표시하는 C 응용 프로그램 컨테이너를 빌드합니다.

1. 빈 디렉터리를 만들고 C 소스 코드 및 Dockerfile을 추가합니다.

```
$ 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. 다음 명령을 실행하여 컨테이너를 빌드합니다.

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

3. 다음 명령을 실행하여 컨테이너를 실행합니다.

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

Amazon Linux 2023 컨테이너 이미지에 설치된 패키지 비교

AL2023 기본 컨테이너 이미지에 있는 RPM과 AL2023 최소 컨테이너 이미지에 있는 RPM 간의 비교입니다.

패키지	컨테이너	최소 컨테이너
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	2023년 2월 68일	2023년 2월 68일
coreutils-single	8.32	8.32
crypto-policies	20220428	20220428
curl-minimal	8.5.0	8.5.0

패키지	컨테이너	최소 컨테이너
dnf	4.14.0	
dnf-data	4.14.0	4.14.0
elfutils-default-yama-scope	0.188	
elfutils-libelf	0.188	
elfutils-libs	0.188	
expat	2.5.0	
file-libs	5.39	5.39
filesystem	3.14	3.14
gawk	5.1.0	5.1.0
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

패키지	컨테이너	최소 컨테이너
grep	3.8	3.8
json-c	0.14	0.14
keyutils-libs	1.6.3	1.6.3
krb5-libs	1.21.3	1.21.3
libacl	2.3.1	2.3.1
libarchive	3.7.4	3.7.4
libassuan	2.5.5	2.5.5
libattr	2.5.1	2.5.1
libblkid	2.37.4	2.37.4
libcap	2.48	2.48
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

패키지	컨테이너	최소 컨테이너
libidn2	2.3.2	2.3.2
libmodulemd	2.13.0	2.13.0
libmount	2.37.4	2.37.4
libnghttp2	1.59.0	1.59.0
libpeas		1.32.0
libpsl	0.21.1	0.21.1
librepo	1.14.5	1.14.5
libreport-filessystem	2.15.2	2.15.2
libselinux	3.4	3.4
libsepol	3.4	3.4
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

패키지	컨테이너	최소 컨테이너
libyaml	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
lua-libs	5.4.4	5.4.4
lz4-libs	1.9.4	1.9.4
microdnf		3.10.0
microdnf-dnf		3.10.0
mpfr	4.1.0	4.1.0
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
npth	1.6	1.6
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	

패키지	컨테이너	최소 컨테이너
python3-hawkey	0.69.0	
python3-libcomps	0.1.20	
python3-libdnf	0.69.0	
python3-libs	3.9.16	
python3-pip-wheel	21.3.1	
python3-rpm	4.16.1.3	
python3-setuptools-wheel	59.6.0	
readline	8.1	8.1
rpm	4.16.1.3	4.16.1.3
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 미니멀 AMI와 컨테이너 이미지에 설치된 패키지 비교

AL2023 최소 AMI에 있는 RPM과 AL2023 기본 및 최소 컨테이너 이미지에 있는 RPM 간의 비교입니다.

패키지	미니멀 AMI	컨테이너	최소 컨테이너
alternatives	1.15	1.15	1.15
amazon-chrony-config	4.3		
amazon-ec2-net-utils	2.5.1		
amazon-linux-repo-cdn		2023.6.20241031	2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031		
amazon-linux-sb-keys	2023.1		
amd-ucode-firmware	20210208(noarch)		
audit	3.0.6		
audit-libs	3.0.6	3.0.6	3.0.6
awscli-2	2.15.30		
basesystem	11	11	11
bash	5.2.15	5.2.15	5.2.15
bzip2-libs	1.0.8	1.0.8	1.0.8
ca-certificates	2023년 2월 68일	2023년 2월 68일	2023년 2월 68일

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		
dbus-common	1.12.28		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
dbus-libs	1.12.28		
device-mapper	1.02.185		
device-mapper-libs	1.02.185		
diffutils	3.8		
dnf	4.14.0	4.14.0	
dnf-data	4.14.0	4.14.0	4.14.0
dnf-plugin-release-notification	1.2		
dnf-plugins-core	4.3.0		
dnf-plugin-support-info	1.2		
dracut	055		
dracut-config-ec2	3.0		
dracut-config-generic	055		
e2fsprogs	1.46.5		
e2fsprogs-libs	1.46.5		
ec2-utils	2.2.0		
efi-filesystem	5		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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	2019년 0월 19일		
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	

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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		

패키지	미니멀 AMI	컨테이너	최소 컨테이너
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 Beanstalk의 AL2023

AWS Elastic Beanstalk은 웹 애플리케이션 및 서비스를 배포하고 확장하는 서비스입니다. 코드를 업로드하면 Elastic Beanstalk가 용량 프로비저닝, 로드 밸런싱, Auto Scaling부터 애플리케이션 상태 모니터링에 이르기까지 자동으로 배포합니다. 자세한 내용은 [AWS Elastic Beanstalk](#)를 참조하세요.

Elastic Beanstalk를 사용하려면 애플리케이션을 생성하고, 애플리케이션 소스 번들 포맷(예: Java .war 파일)으로 애플리케이션 버전을 업로드한 다음, 애플리케이션에 대한 몇 가지 정보를 입력합니다. Elastic Beanstalk로 환경을 실행한 다음 코드 실행에 필요한 AWS 리소스를 생성하고 구성합니다. 자세한 내용은 [AWS Elastic Beanstalk 개발자 안내서](#)를 참조하세요.

Amazon EC2 인스턴스에서 Elastic Beanstalk Linux 플랫폼이 실행되며 이 인스턴스는 Amazon Linux에서 실행됩니다. 2023년 8월 4일 현재, Elastic Beanstalk는 Amazon Linux 2023 기반에서 Docker, Tomcat, Java SE, Node.js, PHP 및 Python 등의 플랫폼 브랜치를 지원합니다. Elastic Beanstalk는 앞으로 더 많은 Elastic Beanstalk 플랫폼에 AL2023을 지원하기 위해 노력하고 있습니다.

Elastic Beanstalk 플랫폼 지원 및 AL2023에 구축된 현재 플랫폼의 전체 목록은 [Elastic Beanstalk 개발자 가이드](#)의 [Elastic Beanstalk 리눅스 플랫폼](#) 섹션에서 확인할 수 있습니다.

최신 Elastic Beanstalk 플랫폼 및 기존 플랫폼 버전에 대한 릴리스 노트는 [Elastic Beanstalk 릴리스 노트](#)에 있습니다.

AWS CloudShell에서 AL2023 사용

AWS CloudShell은 브라우저 기반의 사전 인증된 셸로, AWS Management Console에서 바로 시작할 수 있습니다. AWS Management Console에서 CloudShell로 이동하는 몇 가지 방법이 있습니다. 자세한 내용은 [AWS CloudShell 시작하는 방법](#)을 참조하세요.

현재 Amazon Linux 2 기반인 AWS CloudShell을 AL2023으로 마이그레이션합니다. 전체 AWS 리전에서 AL2023으로의 마이그레이션은 2023년 12월 4일부터 전면 시행될 예정입니다. AL2023으로 마이그레이션하는 CloudShell에 대한 자세한 내용은 [Amazon Linux 2에서 Amazon Linux 2023으로 AWS CloudShell 마이그레이션하기](#)을 참조하세요.

AL2023 기반 Amazon ECS AMI를 사용하여 컨테이너화된 워크로드 호스팅

Note

컨테이너에서 AL2023을 사용하는 방법에 대한 자세한 내용은 [AL2023 컨테이너](#) 섹션을 참조하세요.

Amazon Elastic Container Service(Amazon ECS)는 컨테이너 애플리케이션을 쉽게 배포, 관리 및 확장할 수 있도록 도와주는 완전 관리형 컨테이너 오케스트레이션 서비스입니다. 완전 관리형 서비스인 Amazon ECS에는 AWS 구성과 운영 모범 사례가 내장되어 있습니다. AWS 및 Amazon Elastic Container Registry(Amazon ECR), Docker 등의 서드 파티 도구와도 통합됩니다. 이러한 통합을 통해 환경이 아닌 애플리케이션 구축에 더욱 집중할 수 있습니다. 컨트롤 플레인을 관리하는 복잡한 과정 없이 클라우드의 AWS 리전에서 컨테이너 워크로드를 실행하고 확장할 수 있습니다.

AL2023 기반 Amazon ECS 최적화 AMI를 사용하면 AL2023에서 컨테이너 워크로드를 호스팅할 수 있습니다. 자세한 내용은 [Amazon ECS 최적화 AMI](#) 섹션을 참조하세요.

AL2와 비교한 AL2023 Amazon ECS 변경 사항

AL2와 함께, AL2023은 Amazon ECS Linux 인스턴스로 실행하는 데 필요한 기본 패키지를 제공합니다. AL2에서 amazon-linux-extras를 통해 containerd, docker 및 ecs-init 패키지를 사용할 수 있었지만 AL2023에는 이러한 패키지가 Core 리포지토리에 포함되어 있습니다.

버전이 지정된 리포지토리 기능으로 결정적 업그레이드 기능을 사용하면 모든 AL2023 AMI는 특정 리포지토리 버전에서만 사용할 수 있게 됩니다. 이는 AL2023 Amazon ECS 최적화 AMI에서도 마찬가지입니다. 환경에 대한 모든 업데이트를 배포 전에 주의 깊게 관리하고 테스트할 수 있을 뿐만 아니라 문제 발생 시 이전 AMI 콘텐츠로 쉽게 되돌릴 수 있습니다. 이 AL2023 기능에 대한 자세한 내용은 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#)를 참조하세요.

AL2023 버전은 AL2에서 지원되는 cgroup v1 인터페이스를 통해 cgroup v2로 전환됩니다. 자세한 내용은 [통합 제어 그룹 계층 구조 \(cgroup v2\)](#) 섹션을 참조하세요.

Note

[2023.2.20230920](#) 이전 AL2023 버전(첫 번째 AL2023.2 릴리스)에는 cgroup 내에서 메모리 부족(OOM) 처리를 위한 systemd에 버그가 포함되어 있습니다. cgroup의 모든 프로세스는

OOM 킬러가 한 번에 하나의 프로세스를 선택하는 대신 항상 종료되었으며, 이는 의도한 동작입니다.
이는 AL2 동작과 달리 회귀 현상이었으며, AL2023 2023.2.20230920 릴리스를 기준으로 수정되었습니다.

Amazon ECS 최적화 AMI를 빌드하는 코드는 [amazon-ecs-ami GitHub 프로젝트](#)에서 확인할 수 있습니다. [릴리스 노트](#)에서는 어떤 AL2023 버전이 어떤 Amazon ECS AMI 버전에 매핑되는지 설명합니다.

AL2023 Amazon ECS에 최적화된 AMI 사용자 지정

Important

Amazon ECS 최적화 AL2023 AMI 사용을 권장합니다. 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 [Amazon ECS 최적화 AMI](#) 섹션을 참조하세요.

Amazon ECS에서 사용자 지정 AMI를 생성할 때 사용하는 것과 동일한 빌드 스크립트를 사용할 수 있습니다. 자세한 내용은 [Amazon ECS 최적화 Linux AMI 빌드 스크립트](#) 섹션을 참조하세요.

AL2023에서 Amazon Elastic File System 사용

Amazon Elastic File System(Amazon EFS)은 완전히 탄력적인 서버리스 파일 스토리지를 제공하므로 스토리지 용량과 성능을 프로비저닝하거나 관리하지 않고도 파일 데이터를 공유할 수 있습니다. Amazon EFS는 애플리케이션을 중단하지 않고 페타바이트급까지 온디맨드 규모로 확장할 수 있도록 구축되었으며 파일을 추가하고 제거할 때 확장 및 축소됩니다. Amazon EFS에는 간단한 웹 서비스 인터페이스가 있으므로 파일 시스템을 빠르고 쉽게 생성하고 구성할 수 있습니다. 모든 파일 스토리지 인프라를 관리하는 서비스이므로 사용자는 복잡한 파일 시스템 구성을 배포, 패치 및 유지 보수하는 번잡함에서 벗어날 수 있습니다.

Amazon EFS에서는 Network File System 버전 4(NFSv4.1 및 NFSv4.0) 프로토콜을 지원하므로 오늘날 사용하는 애플리케이션 및 도구도 Amazon EFS에서 원활하게 작동합니다. Amazon EC2, Amazon ECS 및 AWS Lambda 를 비롯한 여러 컴퓨팅 인스턴스가 동시에 Amazon EFS 파일 시스템에 액세스할 수 있습니다. 따라서, EFS 파일 시스템은 다수의 인스턴스 또는 서버에서 실행되는 워크로드와 애플리케이션의 공통 데이터 소스를 제공합니다.

AL2023에 amazon-efs-utils 설치

amazon-efs-utils 패키지는 AL2023 리포지토리에 설치하여 Amazon EFS 파일 시스템에 액세스할 수 있습니다.

AL2023에 **amazon-efs-utils** 패키지를 설치하세요.

- 다음 명령을 사용하여 amazon-efs-utils를 설치합니다.

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

AL2023에 Amazon EFS 파일 시스템 마운트

amazon-efs-utils를 설치했다면 Amazon EFS 파일 시스템을 AL2023 인스턴스에 마운트할 수 있습니다.

AL2023에 Amazon EFS 파일 시스템 마운트

- 다음 명령을 사용하여 파일 시스템 id로 마운트합니다.

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

전송 중인 데이터가 TLS를 사용하거나 파일 시스템 ID 대신 DNS 이름 또는 마운트 대상 IP를 사용하여 암호화되도록 파일 시스템을 마운트할 수도 있습니다. 자세한 내용은 [EFS 마운트 도우미로 Amazon Linux 인스턴스에 마운트](#)를 참조하세요.

AL2023에서 Amazon EMR 사용

Amazon EMR은 Apache Hadoop 및 AWS가 제공하는 서비스를 사용하여 많은 데이터를 쉽고 효율적으로 처리할 수 있도록 지원하는 웹 서비스입니다.

AL2023 기반 Amazon EMR 릴리스

Amazon EMR 릴리스 7.0.0은 AL2023에 빌드된 첫 번째 릴리스였습니다. 이번 릴리스에서 AL2023은 Amazon EMR의 기본 운영 체제로, Amazon EMR에 AL2023의 모든 이점을 제공합니다. 자세한 내용은 [Amazon EMR 7.0.0 릴리스 노트](#)를 참조하세요.

EKS 기반 Amazon EMR에 AL2023 지원

EKS 6.13 기반 Amazon EMR은 AL2023을 옵션으로 도입한 첫 번째 릴리스입니다. 이번 릴리스에서는 Java 17 런타임과 함께 AL2023 운영 체제로 Spark를 실행할 수 있습니다. 자세한 내용은 [EKS 6.13 기반 Amazon EMR 릴리스 노트](#) 및 [EKS 기반 Amazon EMR 릴리스 노트](#)를 참조하세요.

에서 AL2023 사용 AWS Lambda

를 사용하면 서버를 프로비저닝하거나 관리하지 않고도 코드를 실행할 AWS Lambda 수 있습니다. 사용한 컴퓨팅 시간에 대해서만 요금을 지불하면 되고, 코드가 실행되지 않을 때는 요금이 부과되지 않습니다. 관리할 필요 없이 사실상 모든 애플리케이션 또는 백엔드 서비스의 코드를 가상으로 실행할 수 있습니다. 코드를 업로드하기만 하면 고가용성을 유지한 채로 코드를 실행하고 확장하는 데 필요한 모든 것을 Lambda가 알아서 처리해 줍니다.

AL2023 **provided.al2023** 관리형 런타임 및 컨테이너 이미지

provided.al2023 기본 런타임은 [AL2023 최소 컨테이너 이미지](#)를 기반으로 하며 AL2023 기반 Lambda 관리형 런타임 및 [컨테이너 기본 이미지](#)를 제공합니다. provided.al2023 런타임은 AL2023 최소 컨테이너 이미지를 기반으로 하므로 약 109MB의 provided.al2 런타임보다 40MB 미만에서 상당히 작습니다.

자세한 내용은 [Lambda 런타임](#) 및 [Lambda 컨테이너 이미지 작업](#) 섹션을 참조하세요.

AL2023 기반 Lambda 런타임

Node.js 20, Python 3.12, Java 21, .NET 8과 같은 관리형 언어 런타임의 향후 릴리스는 AL2023 기반이며, [AL2023 기반 런타임 공지](#)에 설명된 대로 provided.al2023를 기본 이미지로 사용합니다.

AL2023 기반 Lambda 함수

- [Go에 작성된 AL2023 Lambda 함수](#)
- [Rust에 작성된 AL2023 Lambda 함수](#)

자세한 내용은 [개발자 가이드](#)의 AWS Lambda Lambda 런타임을 참조하세요.

자습서

다음 자습서에서는 Amazon Linux 2023(AL2023)을 실행하는 Amazon EC2 인스턴스로 일반적인 태스크를 수행하는 방법을 보여줍니다. 동영상 자습서는 [AWS 교육 동영상 및 실습](#) 섹션을 참조하세요.

AL2 지침은 Amazon EC2 사용 설명서의 [Linux를 실행하는 Amazon EC2 인스턴스용 자습서](#) 섹션을 참조하세요.

자습서

- [자습서: AL2023에 LAMP 서버 설치](#)
- [자습서: AL2023에서 SSL/TLS 구성](#)
- [자습서: AL2023에서 WordPress 블로그 호스트](#)
- [자습서: Redis 6에서 AL2023의 Valkey로 전환](#)
- [자습서: AL2023에 GNOME 데스크톱 환경 설치](#)
- [자습서: AL2023에서 TigerVNC 서버 구성](#)
- [AL2023 커널에서 Multi-Gen LRU\(MGLRU\) 사용](#)
- [자습서: AL2023에 LibreOffice 설치](#)

자습서: AL2023에 LAMP 서버 설치

다음 절차를 통해 AL2023 인스턴스에서 PHP 및 [MariaDB](#)(커뮤니티에서 개발한 MySQL 포크) 지원을 포함하는 Apache 웹 서버를 설치할 수 있습니다. 이 웹 서버를 LAMP 웹 서버 또는 LAMP 스택이라고도 합니다. 이 서버를 사용해서 고정 웹사이트를 호스팅하거나 데이터베이스에서 정보를 읽고 쓰는 동적 PHP 애플리케이션을 배포할 수 있습니다.

Important

이 절차는 AL2023에서 사용하기 위한 것입니다. Ubuntu 또는 Red Hat Enterprise, Linux와 같은 다른 배포에서 LAMP 웹 서버를 설정하려는 경우 이 자습서를 이용할 수 없습니다. Ubuntu의 경우 Ubuntu 커뮤니티 설명서 [ApacheMySQLPHP](#)를 참조하세요. 다른 배포는 관련 설명서를 참조하세요.

작업

- [1단계: LAMP 서버 준비](#)
- [2단계: LAMP 서버 테스트](#)
- [3단계: 데이터베이스 서버 보안 설정](#)
- [4단계: \(선택 사항\) phpMyAdmin 설치](#)
- [문제 해결](#)
- [관련 주제](#)

1단계: LAMP 서버 준비

사전 조건

- 본 자습서는 AL2023을 사용하여 사용자가 인터넷에서 접근할 수 있는 퍼블릭 DNS 이름을 가진 새 인스턴스를 이미 실행한 것으로 가정하고 있습니다. 자세한 내용은 [Amazon EC2 AL2023](#) 단원을 참조하십시오. SSH(포트 22), HTTP(포트 80), HTTPS(포트 443) 연결을 허용할 수 있도록 보안 그룹을 구성해야 합니다. 이러한 전제 조건에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Linux 인스턴스의 인바운드 트래픽 권한 부여](#) 섹션을 참조하세요.
- 다음 절차는 AL2023에서 사용할 수 있는 최신 PHP 버전(현재 8.1)을 설치합니다. 이 자습서에서 명시한 애플리케이션이 아닌 PHP 애플리케이션을 사용하려는 경우 8.1과의 호환성을 확인해야 합니다.

LAMP 서버를 준비하려면

1. 인스턴스에 연결합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 단원을 참조하십시오.
2. 모든 소프트웨어 패키지가 최신 상태로 업데이트되어 있는지 확인하기 위해, 인스턴스에서 쿼크 소프트웨어 업데이트를 실행합니다. 이 과정은 몇 분 정도 시간이 소요될 수 있지만, 최신 보안 업데이트와 버그 수정이 있는지 확인하는 것이 중요합니다.

-y 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

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

3. 최신 버전의 Apache 웹 서버와 AL2023용 PHP 패키지를 설치합니다.

```
[ec2-user ~]$ sudo dnf install -y httpd wget php-fpm php-mysql php-json php php-devel
```


4. 소프트웨어 패키지 MariaDB를 설치합니다. `dnf install` 명령을 사용하여 여러 소프트웨어 패키지와 모든 관련 종속 프로그램을 동시에 설치합니다.

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

다음 명령을 사용하여 이러한 패키지의 현재 버전을 볼 수 있습니다.

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

예제:

```
[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 웹 서버를 시작합니다.

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

6. `systemctl` 명령을 사용하여 Apache 웹 서버가 매번 시스템이 부팅할 때마다 시작되도록 합니다.

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

다음 명령을 실행하여 httpd가 실행되고 있는지 확인할 수 있습니다.

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

7. 인스턴스에 대해 인바운드 HTTP(포트 80) 연결을 허용하는 보안 규칙이 없는 경우 추가합니다. 기본적으로 시작하는 동안 인스턴스에 대해 **launch-wizard-N** 보안 그룹이 생성되었습니다. 보안 그룹 규칙을 추가하지 않은 경우 이 그룹에는 SSH 연결을 허용하는 단일 규칙만 포함됩니다.

- a. <https://console.aws.amazon.com/ec2/>에서 Amazon EC2 콘솔을 엽니다.
- b. 탐색 창에서 인스턴스(Instances)를 선택하고 인스턴스를 선택합니다.
- c. 보안 탭에서 인바운드 규칙을 확인합니다. 다음과 같은 규칙이 표시되어야 합니다.

Port range	Protocol	Source
22	tcp	0.0.0.0/0

Warning

0.0.0.0/0을 사용하면 모든 IP 주소에서 SSH를 사용하여 인스턴스에 액세스할 수 있습니다. 테스트 환경에서 잠시 사용하는 것은 괜찮지만 프로덕션 환경에서는 안전하지 않습니다. 프로덕션에서는 특정 IP 주소나 주소 범위만 인스턴스에 액세스하도록 허용하세요.

- d. HTTP(포트 80) 연결을 허용하는 인바운드 규칙이 없는 경우 지금 규칙을 추가해야 합니다. 보안 그룹에 대한 링크를 선택합니다. [Linux 인스턴스의 인바운드 트래픽 권한 부여](#) 섹션의 절차를 사용하여 다음 값으로 새로운 인바운드 보안 규칙을 추가합니다.
 - 유형: HTTP
 - 프로토콜: TCP
 - 포트 범위: 80
 - 소스: 사용자 지정
8. 웹 서버를 테스트합니다. 웹 브라우저에서 인스턴스의 공용 DNS 주소(또는 공용 IP 주소)를 입력합니다. /var/www/html에 콘텐츠가 없으면 "It works!"라는 메시지를 보여 주는 Apache 테스트 페이지가 표시됩니다.

Amazon EC2 콘솔을 사용하여 인스턴스에 대한 퍼블릭 DNS를 가져올 수 있습니다(퍼블릭IPv4 DNS 열 확인, 이 열이 숨겨진 경우 기본 설정(기어 모양 아이콘)을 선택하고 퍼블릭 IPv4 DNS를 켵니다).

인스턴스의 보안 그룹에 포트 80에서 HTTP 트래픽을 허용하는 규칙이 포함되어 있는지 확인합니다. 자세한 내용은 [보안 그룹 규칙 추가](#) 섹션을 참조하세요.

Important

Amazon Linux를 사용하지 않는 경우, 이러한 연결을 허용하도록 인스턴스에서 방화벽을 구성해야 할 수도 있습니다. 방화벽 구성 방법에 대한 자세한 내용은 사용자의 특정 배포에 대한 문서를 참조하세요.

Apache httpd는 Apache document root라는 디렉터리에 보관된 파일을 처리합니다. Amazon Linux Apache 문서 루트는 /var/www/html이며, 기본적으로 루트에서 소유합니다.

ec2-user 계정에서 이 디렉터리의 파일을 조작할 수 있게 하려면 디렉터리의 소유권과 권한을 변경해야 합니다. 이 작업을 수행하는 방법에는 여러 가지가 있습니다. 본 자습서에서는 ec2-user를 apache 그룹에 추가하여 apache 그룹에 /var/www 디렉터리의 소유권을 부여하고 쓰기 권한을 할당합니다.

파일 권한 설정

1. 사용자(이 경우는 ec2-user)를 apache 그룹에 추가합니다.

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

2. 로그아웃하고 다시 로그인한 다음, 새 그룹을 선택하고 멤버십을 확인합니다.

- a. 로그아웃합니다(exit 명령을 사용하거나 터미널 창 닫기).

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

- b. apache 그룹의 멤버십을 확인하려면 인스턴스에 다시 연결한 후 다음 명령을 실행합니다.

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

3. /var/www 및 그 콘텐츠의 그룹 소유권을 apache 그룹으로 변경합니다.

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

4. 그룹 쓰기 권한을 추가하여 나중에 하위 디렉터리에 대한 그룹 ID를 설정하려면 /var/www와 그 하위 디렉터리의 디렉터리 권한을 변경합니다.

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

5. 그룹 쓰기 권한을 추가하려면 /var/www 및 그 하위 디렉터리의 파일 권한을 반복하여 변경합니다.

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

이제 ec2-user와 apache 그룹의 향후 멤버는 Apache document root에서 파일 추가, 삭제, 편집을 할 수 있고, 이를 통해 사용자는 정적 웹 사이트 또는 PHP 애플리케이션과 같은 콘텐츠를 추가할 수 있습니다.

웹 서버를 보호하려면(선택 사항)

HTTP 프로토콜을 실행하는 웹 서버는 송신하거나 수신하는 데이터에 대해 아무런 전송 보안 기능도 제공하지 않습니다. 웹 브라우저를 사용하여 HTTP 서버에 연결할 때 방문하는 URL, 수신하는 웹 페이지의 내용, 제출하는 HTML 양식의 내용(암호 포함)이 모두 네트워크 경로를 따라 어디서든 엿보려는 사람들에게 보입니다. 웹 서버를 안전하게 보호하기 위한 최선의 방법은 SSL/TLS 암호화로 데이터를 보호하는 HTTPS(HTTP Secure) 지원 기능을 설치하는 것입니다.

서버에서 HTTPS를 활성화하는 방법에 대한 자세한 내용은 [자습서: AL2023에서 SSL/TLS 구성](#) 섹션을 참조하세요.

2단계: LAMP 서버 테스트

서버가 설치되어 실행 중이고 파일 권한이 올바르게 설정되었다면 사용자의 ec2-user 계정을 통해 인터넷에서 사용 가능한 /var/www/html 디렉터리에서 PHP 파일을 생성할 수 있어야 합니다.

LAMP 서버를 테스트하려면

1. Apache 문서 루트에서 PHP 파일을 생성합니다.

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

이 명령을 실행하는 동안 "Permission denied" 오류가 발생하면, 로그아웃하고 다시 로그인한 다음, [파일 권한 설정](#)에서 구성한 적절한 그룹 권한을 선택합니다.

- 웹 브라우저에서는 방금 생성한 파일의 URL을 입력합니다. 이 URL은 인스턴스의 퍼블릭 DNS 주소에 슬래시(/)와 파일 이름이 추가된 형태입니다. 다음 예를 참조하세요.

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

PHP 정보 페이지가 표시되어야 합니다:

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-mysqld.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



이 페이지가 보이지 않을 경우 이전 단계에서 `/var/www/html/phpinfo.php` 파일이 제대로 생성되었는지 확인하세요. 또한 다음 명령을 사용하여 필수 패키지가 모두 설치되었는지도 확인할 수 있습니다.

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

출력에서 필요한 패키지가 하나라도 나열되지 않으면, `sudo yum install package` 명령을 사용하여 패키지를 설치합니다.

3. `phpinfo.php` 파일을 삭제합니다. 이 파일은 유용한 정보를 포함하고 있지만 보안상 이유로 인터넷에 공개되어서는 안 됩니다.

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

이제 LAMP 웹 서버가 완전히 동작하는 상태가 됩니다. `/var/www/html`의 Apache document root에 콘텐츠를 추가하면 인스턴스에 대한 퍼블릭 DNS 주소에서 그 콘텐츠를 볼 수 있습니다.

3단계: 데이터베이스 서버 보안 설정

MariaDB 서버의 기본 설치에는 테스트 및 개발 기능에 유용한 여러 기능이 포함되어 있지만, 이 기능들은 프로덕션 서버에서는 비활성화되거나 제거되어야 합니다. `mysql_secure_installation` 명령을 통해 루트 암호를 설정하고 설치 패키지에서 보안성이 낮은 기능을 제거하는 과정을 수행할 수 있습니다. MariaDB 서버를 사용할 계획이 없더라도 이 절차를 수행하는 것이 좋습니다.

MariaDB 서버의 보안을 유지하려면

1. MariaDB 서버를 시작합니다.

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

2. `mysql_secure_installation`를 실행합니다.

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

- a. 암호를 입력하라는 메시지가 표시되면 루트 계정의 암호를 입력합니다.
 - i. 현재 루트 암호를 입력합니다. 기본적으로 root 계정에는 암호가 없습니다. Enter를 누릅니다.

- ii. 암호를 설정하려면 **Y**를 누른 후 안전한 암호를 두 번 입력합니다. 보안 암호 생성에 대한 자세한 내용은 <https://identitysafe.norton.com/password-generator/> 섹션을 참조하세요. 이 암호를 안전한 장소에 보관하시기 바랍니다.

MariaDB에 대한 루트 암호를 설정하는 것은 데이터베이스를 보호하는 가장 기초적인 방법일 뿐입니다. 데이터베이스 기반 애플리케이션을 빌드하거나 설치할 때, 일반적으로 그 애플리케이션의 데이터베이스 서비스 사용자를 만들고 데이터베이스 관리 이외의 어떤 목적으로도 루트 계정을 사용하지 못하게 합니다.

- b. **Y**를 눌러서 익명 사용자 계정을 제거합니다.
 - c. **Y**를 입력하여 원격 루트 로그인을 비활성화합니다.
 - d. **Y**를 눌러서 테스트 데이터베이스를 제거합니다.
 - e. **Y**를 눌러서 권한 테이블을 다시 로드하고 변경사항을 저장합니다.
3. (선택 사항) 지금 바로 사용할 계획이 아니라면 MariaDB 서버를 중지합니다. 필요할 때 다시 시작할 수 있습니다.

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

4. (선택 사항) 부팅 시 MariaDB 서버가 시작되도록 하려면 다음 명령을 입력합니다.

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

4단계: (선택 사항) phpMyAdmin 설치

[phpMyAdmin](#)은 EC2 인스턴스의 MySQL 데이터베이스를 보고 편집하는 데 사용할 수 있는 웹 기반 데이터베이스 관리 도구입니다. Amazon Linux 인스턴스에서 phpMyAdmin을 설치 및 구성하려면 다음 단계를 따르세요.

Important

Apache에서 SSL/TLS를 활성화하지 않은 경우 phpMyAdmin을 사용하여 LAMP 서버에 액세스하지 않는 것이 좋습니다. 이 상태에서 액세스하면 데이터베이스 관리자 암호와 기타 데이터가 인터넷을 통해 안전하지 못한 상태로 전송됩니다. 개발자의 보안 권장 사항을 보려면 [phpMyAdmin 설치 보안](#)을 참조하세요. EC2 인스턴스에서의 웹 서버 보안에 대한 일반적인 정보는 [자습서: AL2023에서 SSL/TLS 구성](#) 섹션을 참조하세요.

phpMyAdmin을 설치하려면

1. 필요한 종속 항목을 설치합니다.

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

2. Apache를 다시 시작합니다.

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

3. php-fpm을 다시 시작합니다.

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

4. Apache 문서 루트(/var/www/html)로 이동합니다.

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

5. <https://www.phpmyadmin.net/downloads>에서 phpMyAdmin 최신 릴리스의 소스 패키지를 선택합니다. 인스턴스로 파일을 직접 다운로드하려면 다음 예제와 같이 링크를 복사한 후 wget 명령에 붙여 넣습니다.

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

6. phpMyAdmin 폴더를 생성하고 다음 명령을 사용하여 해당 폴더로 패키지의 압축을 풉니다.

```
[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* tarball을 삭제합니다.

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

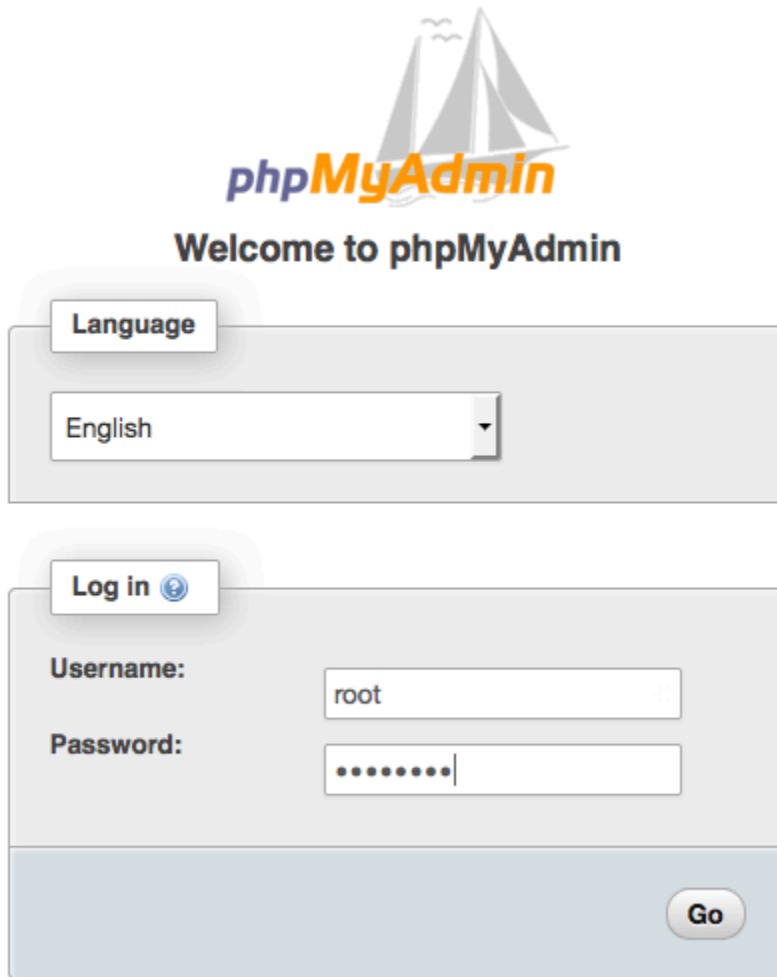
8. (선택 사항) MySQL 서버가 실행 중이지 않으면 지금 시작합니다.

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

9. 웹 브라우저에서 phpMyAdmin 설치의 URL을 입력합니다. 아래의 예와 같이 이 URL은 인스턴스의 퍼블릭 DNS 주소(또는 퍼블릭 IP 주소)에 슬래시(/)와 설치 디렉터리의 이름이 추가된 형태입니다. 다음 예를 참조하세요.

<http://my.public.dns.amazonaws.com/phpMyAdmin>

사용자는 phpMyAdmin 로그인 페이지를 볼 수 있어야 합니다:



10. 앞서 만든 root 사용자 이름 및 MySQL 루트 암호로 phpMyAdmin 설치에 로그인합니다.

작동하려면 먼저 설치를 구성해야 합니다. 먼저 다음과 같이 구성 파일을 수동으로 작성하는 것이 좋습니다.

- 최소 구성 파일로 시작하려면 자주 사용하는 텍스트 편집기를 사용하여 새 파일을 생성한 후에 `config.sample.inc.php` 내용을 파일에 복사합니다.
- 이 파일을 `config.inc.php`가 포함된 phpMyAdmin 디렉토리에 `index.php`로 저장하세요.
- 추가 설정에 대해서는 phpMyAdmin 설치 지침에서 [설정 스크립트 사용](#) 섹션의 파일 생성 후 지침을 참조하세요.

phpMyAdmin에 대한 자세한 내용은 [phpMyAdmin 사용 설명서](#)를 참조하세요.

문제 해결

이 섹션에서는 새 LAMP 서버를 설정하는 동안 발생할 수 있는 일반적인 문제 해결을 위한 제안을 제공합니다.

웹 브라우저를 사용하여 서버에 연결할 수 없음

다음을 확인하여 Apache 웹 서버가 실행 중이고 액세스 가능한지 확인합니다.

- 웹 서버가 실행되고 있습니까?

다음 명령을 실행하여 httpd가 실행되고 있는지 확인할 수 있습니다.

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

httpd 프로세스가 실행되지 않는 경우 [LAMP 서버를 준비하려면](#)에 설명된 단계를 반복합니다.

- 방화벽이 올바르게 구성되었습니까?

인스턴스의 보안 그룹에 포트 80에서 HTTP 트래픽을 허용하는 규칙이 포함되어 있는지 확인합니다. 자세한 내용은 [보안 그룹 규칙 추가](#) 섹션을 참조하세요.

HTTPS를 사용하여 서버에 연결할 수 없음

다음 확인을 수행하여 Apache 웹 서버가 HTTPS를 지원하도록 구성되어 있는지 확인합니다.

- 웹 서버가 올바르게 구성되었습니까?

Apache를 설치한 후 서버는 HTTP 트래픽에 대해 구성됩니다. HTTPS를 지원하려면 서버에서 TLS를 활성화하고 SSL 인증서를 설치합니다. 자세한 내용은 [자습서: AL2023에서 SSL/TLS 구성](#) 단원을 참조하세요.

- 방화벽이 올바르게 구성되었습니까?

인스턴스의 보안 그룹에 포트 443에서 HTTP 트래픽을 허용하는 규칙이 포함되어 있는지 확인합니다. 자세한 내용은 [Linux 인스턴스의 인바운드 트래픽 권한 부여](#) 섹션을 참조하세요.

관련 주제

파일을 인스턴스에 전송하거나 웹 서버에 WordPress 블로그를 설치하는 것에 대한 자세한 내용은 다음 문서를 참고하세요.

- Amazon EC2 사용 설명서의 [WinSCP를 사용하여 Linux 인스턴스로 파일 전송](#).
- Amazon EC2 사용 설명서의 [SCP 클라이언트를 사용하여 Linux 인스턴스로 파일 전송](#).
- [자습서: AL2023에서 WordPress 블로그 호스트](#)

이 자습서에서 사용되는 명령과 소프트웨어에 대한 자세한 내용은 다음 웹 페이지를 확인하세요.

- Apache 웹 서버: <http://httpd.apache.org/>
- MariaDB 데이터베이스 서버: <https://mariadb.org/>
- PHP 프로그래밍 언어: <http://php.net/>

웹 서버에 대한 도메인 이름을 등록하거나 기존 도메인 이름을 현재 호스트로 이전하는 것에 대한 자세한 내용은 Amazon Route 53 개발자 안내서의 [Amazon Route 53에서 도메인 및 하위 도메인 생성 및 마이그레이션](#)을 참조하세요.

자습서: AL2023에서 SSL/TLS 구성

Secure Sockets Layer/Transport Layer Security(SSL/TLS)는 웹 서버와 웹 클라이언트 간 암호화된 채널을 만들어 전송 중인 데이터가 도청되지 않도록 보호합니다. 이 자습서에서는 Apache 웹 서버 및 AL2023의 EC2 인스턴스에 SSL/TLS 지원을 수동으로 추가하는 방법을 설명합니다. 이 자습서에서는 로드 밸런서를 사용하고 있지 않다고 가정합니다. Elastic Load Balancing을 사용하는 경우 [AWS Certificate Manager](#)의 인증서를 대신 사용하여 로드 밸런서에 SSL 오프로드를 구성하도록 선택할 수 있습니다.

일반적으로 웹 암호화를 단순히 SSL이라고 부릅니다. 웹 브라우저에서 여전히 SSL을 지원하지만, 후속 프로토콜인 TLS가 공격에 덜 취약합니다. AL2023은 기본적으로 모든 버전의 SSL을 서버 측에서 비활성화합니다. [보안 표준 본문](#)에서는 TLS 1.0이 안전하지 않다고 간주합니다. TLS 1.0 및 TLS 1.1은 2021년 3월에 공식적으로 [사용 중단](#)되었습니다. 이 자습서에서는 TLS 1.2 활성화만을 기반으로 하여 지침을 제공합니다. TLS 1.3은 2018년에 완료되었으며, 기본 TLS 라이브러리(이 자습서의 OpenSSL)가 지원 및 사용 설정되는 한 AL2에서 사용할 수 있습니다. [클라이언트는 2023년 6월 28일 까지 TLS 1.2 이상을 지원해야 합니다](#). 업데이트된 암호화 표준에 대한 자세한 내용은 [RFC 7568](#) 및 [RFC 8446](#)을 참조하세요.

이 자습서는 현대 웹 암호화를 단순히 TLS로 언급합니다.

Important

이 절차는 AL2023에서 사용하기 위한 것입니다. 다른 배포를 실행하는 EC2 인스턴스 또는 이전 Amazon Linux 버전을 실행하는 인스턴스를 설정하려는 경우 이 자습서의 일부 절차가 적용하지 않을 수 있습니다. Ubuntu의 경우 Ubuntu 커뮤니티 설명서: [Ubuntu에서 SSL 열기](#)를 참조하십시오. Red Hat Enterprise Linux의 경우 [Apache HTTP 웹 서버 설정](#)을 참조하세요. 다른 배포는 관련 설명서를 참조하세요.

Note

또는 AWS Nitro Enclaves를 사용하여 Amazon EC2 인스턴스에서 실행되는 웹 애플리케이션 및 서버에 퍼블릭 및 프라이빗 SSL/TLS 인증서를 사용할 수 있는 엔클레이브 애플리케이션인 AWS Nitro 엔클레이브에 AWS Certificate Manager (ACM)을 사용할 수 있습니다. Nitro Enclaves는 격리된 컴퓨팅 환경을 생성하여 SSL/TLS 인증서 및 프라이빗 키와 같은 매우 민감한 데이터를 보호하고 안전하게 처리할 수 있도록 지원하는 Amazon EC2 기능입니다. Nitro Enclaves용 ACM은 Amazon EC2 Linux 인스턴스에서 실행되는 nginx와 함께 작동하여 프라이빗 키를 생성하고, 인증서와 프라이빗 키를 배포하며, 인증서 갱신을 관리합니다. Nitro Enclaves용 ACM을 사용하려면 엔클레이브 지원 Linux 인스턴스를 사용해야 합니다. 자세한 내용은 [AWS Nitro Enclaves 사용 설명서의 Nitro Enclaves란 무엇입니까?](#) AWS 및 Nitro Enclaves용 섹션을 참조하세요. [AWS Certificate Manager](#)

내용

- [사전 조건](#)
- [1단계: 서버에서 TLS 활성화](#)
- [2단계: CA가 서명한 인증서 가져오기](#)
- [3단계: 보안 구성 테스트 및 하드닝](#)
- [문제 해결](#)

사전 조건

이 자습서를 시작하기 전에 다음 단계를 완료합니다.

- EBS 지원 AL2023 인스턴스를 시작합니다. 자세한 내용은 [Amazon EC2 AL2023](#) 단원을 참조하십시오.
- 인스턴스가 다음 TCP 포트에서 연결을 허용하도록 보안 그룹을 구성합니다.
 - SSH(포트 22)
 - HTTP(포트 80)
 - HTTPS(포트 443)

자세한 내용은 Amazon EC2 사용 설명서의 [Linux 인스턴스의 인바운드 트래픽 권한 부여](#) 섹션을 참조하세요.

- Apache 웹 서버를 설치합니다. 단계별 지침은 [자습서: AL2023에 LAMP 서버 설치](#) 섹션을 참조하십시오. httpd 패키지와 그 종속 프로그램만 필요합니다. PHP 및 MariaDB와 관련된 지침은 무시해도 됩니다.
- 웹 사이트를 식별하고 인증하려면 TLS 퍼블릭 키 인프라(PKI)는 도메인 이름 시스템(DNS)을 사용합니다. EC2 인스턴스를 사용하여 퍼블릭 웹 사이트를 호스팅하려는 경우, 웹 서버의 도메인 이름을 등록하거나 Amazon EC2 호스트로 기존 도메인 이름을 전송해야 합니다. 수많은 타사 도메인 등록 및 DNS 호스팅 서비스를 이에 사용할 수 있습니다. 또는 [Amazon Route 53](#)을 사용할 수도 있습니다.

1단계: 서버에서 TLS 활성화

이 절차에서는 자체 서명된 디지털 인증서를 사용하여 AL2023에 TLS를 설치하는 과정을 보여 줍니다.

Note

자체 서명된 인증서는 테스트에는 허용되지만 프로덕션에는 허용되지 않습니다. 자체 서명된 인증서를 인터넷에 노출하면 사이트 방문자에게 인사말로 보안 경고가 표시됩니다.

서버에서 TLS를 활성화하려면

1. 인스턴스에 연결한 다음 Apache가 실행되는지 확인합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 단원을 참조하십시오.

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

반환된 값이 "enabled"가 아닌 경우 Apache를 시작한 다음 시스템 부팅 시마다 시작하도록 설정합니다.

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

- 모든 소프트웨어 패키지가 최신 상태로 업데이트되어 있는지 확인하기 위해, 인스턴스에서 쿼크 소프트웨어 업데이트를 실행합니다. 이 업데이트 과정은 몇 분 정도 시간이 소요될 수 있지만, 최신 보안 업데이트와 버그 수정을 위해 수행할 필요가 있습니다.

Note

-y 옵션을 사용하면 확인 여부를 묻지 않고 업데이트를 설치합니다. 설치 전에 업데이트 정보를 확인하려면 이 옵션을 생략합니다.

```
[ec2-user ~]$ sudo dnf install openssl mod_ssl
```

- 다음 명령을 입력하면 사이트에 대한 정보를 입력할 수 있는 프롬프트가 표시됩니다.

```
[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
```

이렇게 하면 두 개의 파일이 생성됩니다.

- /etc/pki/tls/private/apache-selfsigned.key (프라이빗 키)
- /etc/pki/tls/certs/apache-selfsigned.crt (자체 서명된 인증서)

인증서 파일 이름은의 SSLCertificateFile 명령에 할당된 기본값과 일치합니다/etc/httpd/conf.d/ssl.conf.

이제 인스턴스에는 보안 서버 구성과 테스트를 위한 인증서 생성에 사용할 다음 파일이 포함됩니다.

- /etc/httpd/conf.d/ssl.conf

mod_ssl의 구성 파일입니다. 여기에는 Apache에 암호화 키 및 인증서의 위치, 허용하는 TLS 프로토콜 버전, 허용하는 암호화 암호를 알려주는 명령이 포함되어 있습니다. 다음은 로컬 인증서 파일입니다.

- /etc/pki/tls/certs/apache-selfsigned.crt

- /etc/pki/tls/private/apache-selfsigned.key

.crt 파일에는 자체 서명된 인증서가 포함되어 있고 .key 파일에는 프라이빗 키가 포함되어 있습니다. Apache에서는 인증서와 키를 PEM 형식으로 요구합니다. 이 형식은 아래의 축약된 예제와 같이 "BEGIN" 및 "END" 라인으로 프레임 처리된 Base64 인코딩 ASCII 문자로 구성됩니다.

```
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBAgwgaggSkAgEAAoIBAQD2KKx/8Zk94m1q
3gQMZF9ZN66Ls19+3tHAgQ5Fpo9KJDhzLj00CI8u1PTcGmAah5kEitCEc0wzmNeo
BC10wYR6G0rGaKtK9Dn7CuIjvubtUysVyQoMVPQ971deakHWeRMiEJFXg6kZZ0vr
GvwnKoMh3D1K44D9dX7IDua2P1Yx5+eroA+1Lqf32ZSaA00bBIMIYTHigwbHMZoT
...
56tE7THvH7v0Ef4/iU0sIrEzaMaJ0mqkmY1A70qQGQKBgBF3H1qNRNHuyMcP0DFs
27hDzPDinrquSEvoZiggkDM1h2irTiipJ/GhkvTpoQlv0fK/VXw8vSgeaBuhwJvS
LXU9HvYq0U604FgD3nAyB9hI0BE13r1HjUvbjT7moH+RhNz6eqqdsCS09VtRAO
4QQvAq0a8UheYeoXLdWcHaLP
-----END PRIVATE KEY-----

-----BEGIN CERTIFICATE-----
MIIEAzCCA10gAwIBAgICWxQwDQYJKoZIhvcNAQELBQAwggbExCzAJBgNVBAYTAi0t
MRIwEAYDVQQIDAlTb21lU3RhdGUxETAPBgNVBACMFNvbWVkaXR5MRkwFwYDVQQK
DBBTb21lT3JnYW5pemF0aW9uMR8wHQYDVQQLDBZTb21lT3JnYW5pemF0aW9uYXV
bm10MRkwFwYDVQQDDDBBpcC0xNzItMzEtMjAtMjMMSQwIgYJKoZIhvcNAQkBFhVy
...
z5rRUE/XzxRLBZ0oWZpNWTXJkQ3uFYH6s/sBwtHpKKZMz0vDedREjNKAvk4ws6F0
CuIjvubtUysVyQoMVPQ971deakHWeRMiEJFXg6kZZ0vrGvwnKoMh3D1K44D9d1U3
WanXWehT6FiSZvB4sTEXXJN2jdw8g+sHGnZ8zC0sc1knYhHrCVD2vnB1ZJKSZvak
3ZazhBxtQSukFM0nWPP2a0DMMFGYUHOd0BQE8sBJxg==
-----END CERTIFICATE-----
```

파일 이름과 확장명은 편의상 사용되며 기능에 영향을 미치지 않습니다. 예를 들어 ssl.conf 파일에서 관련 명령에 동일한 이름을 사용하는 한, 인증서 이름을 cert.crt, cert.pem 또는 다른 파일 이름으로 지정할 수 있습니다.

Note

기본 TLS 파일을 고유의 사용자 지정 파일로 대체하는 경우 파일이 PEM 형식인지 확인하세요.

4. Apache를 다시 시작합니다.

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

Note

앞에서 설명한 대로 EC2 인스턴스에서 TCP 포트 443에 액세스할 수 있는지 확인하세요.

5. Apache 웹 서버가 현재 포트 443에 대해 HTTPS(보안 HTTP)를 지원해야 합니다. 접두사가 **https://**인 브라우저 URL 표시줄에 IP 주소 또는 EC2 인스턴스의 정규화된 도메인 이름을 입력하여 이를 테스트합니다.

신뢰할 수 없는 자체 서명된 호스트 인증서를 사용하여 사이트에 연결하기 때문에 브라우저에 보안 경고가 연속으로 표시될 수 있습니다. 경고를 무시하고 계속 진행합니다.

Apache 기본 테스트 페이지가 열리면 서버에 TLS가 구성되었다는 것입니다. 브라우저와 서버 사이를 통과하는 모든 데이터가 이제 암호화됩니다.

Note

사이트 방문자에게 경고 화면이 표시되는 것을 방지하려면 암호화뿐만 아니라 해당 사이트의 소유자라는 것을 공개적으로 인증하는 신뢰할 수 있는 CA 서명 인증서를 가져와야 합니다.

2단계: CA가 서명한 인증서 가져오기

CA가 서명한 인증서를 가져오려면 다음 절차를 사용할 수 있습니다.

- 프라이빗 키에서 인증서 서명 요청(CSR)을 생성합니다.
- 인증 기관(CA)에 CSR 제출
- 단일 호스트 인증서 가져오기
- Apache를 수정하여 인증서 사용

자체 서명된 TLS X.509 호스트 인증서는 CA가 서명한 인증서와 암호적으로 동일합니다. 그 차이는 수학적인 것이 아니라 사회적입니다. CA는 신청자에게 인증서를 발급하기 전에 도메인의 소유권을 최소한으로 검사합니다. 각 웹 브라우저에는 이를 하도록 브라우저 공급업체에서 신뢰한 CA 목록이 포함되어 있습니다. X.509 인증서는 프라이빗 서버 키에 해당하는 퍼블릭 키와 퍼블릭 키에 암호화 방식으로

연결된 CA의 서명으로 주로 구성되어 있습니다. 브라우저가 HTTPS를 통해 웹 서버에 연결되면 서버는 브라우저에서 신뢰할 수 있는 CA 목록을 확인하도록 인증서를 제공합니다. 서명자가 목록에 있거나 신뢰할 수 있는 다른 서명자로 구성되는 신뢰 체인을 통해 서명자에 액세스할 수 있는 경우, 브라우저는 서버와 암호화된 빠른 데이터 채널을 협상하고 페이지를 로드합니다.

요청 확인 절차로 인해 인증서에는 일반적으로 비용이 발생하므로 여러 인증 기관을 알아봐야 합니다. 일부 CA는 기본 수준 인증서를 무료로 제공합니다. 이러한 CA 중 가장 주목할 만한 것은 [Let's Encrypt](#) 프로젝트인데, 이것은 인증서 생성 및 갱신 프로세스의 자동화도 지원합니다. Let's Encrypt 인증서 사용에 대한 자세한 내용은 [Certbot 받기](#)를 참조하세요.

상업용 서비스를 제공할 계획이라면 [AWS Certificate Manager](#)가 좋은 옵션입니다.

호스트 인증서의 기본을 이루는 것은 키입니다. 2019년 현재 [정부](#) 및 [산업](#) 그룹에서는 2030년까지 문서를 보호하기 위해 마련된 RSA 키에 대해 최소 2048비트의 키(모듈러스) 크기를 사용할 것을 권장합니다. AL2023의 OpenSSL에서 생성된 기본 모듈러스 크기는 2048비트이므로 CA가 서명한 인증서에 사용할 수 있습니다. 다음 절차는 사용자 지정된 키(예: 더 큰 모듈러스 또는 다른 암호화 알고리즘 사용)를 원하는 사용자를 위해 제공되는 선택적 단계입니다.

Important

이 CA 서명 호스트 인증서 획득 지침은 등록과 호스팅이 완료된 DNS 도메인을 소유하지 않을 경우 제대로 적용하기 어렵습니다.

CA가 서명한 인증서를 가져오려면

1. 인스턴스에 연결한 다음 `/etc/pki/tls/private/`으로 이동합니다. 여기는 TLS에 대한 서버의 프라이빗 키를 저장하는 디렉터리입니다. 기존 호스트 키를 사용하여 CSR을 생성하려면 3단계로 건너뛰니다. 인스턴스 연결에 대한 자세한 내용은 [AL2023 인스턴스에 연결](#) 섹션을 참조하세요.
2. (선택 사항) 새 프라이빗 키를 생성합니다. 다음은 몇 가지 키 구성 샘플입니다. 어떤 결과 키도 웹 서버에서 사용할 수 있지만 보안 구현의 정도와 유형은 각각 다릅니다.
 - 예 1: 기본 RSA 호스트 키를 만듭니다. 결과 파일인 **custom.key**는 2048비트 RSA 프라이빗 키입니다.

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key
```

- 예 2: 더 큰 모듈러스로 더 강력한 RSA 키를 만듭니다. 결과 파일인 **custom.key**는 4096비트 RSA 프라이빗 키입니다.

```
[ec2-user ~]$ sudo openssl genrsa -out custom.key 4096
```

- 예 3: 암호로 보호되는 4096비트 암호화 RSA 키를 생성합니다. 그러면 AES-128 암호화로 암호화된 4096비트 RSA 프라이빗 키인 **custom.key** 파일이 생성됩니다.

Important

키 암호화를 통해 보안을 강화할 수 있지만, 암호화된 키에는 암호가 필요하기 때문에 이를 사용하는 서비스는 자동으로 시작할 수 없습니다. 이 키를 사용할 때마다 SSH 연결을 통해 암호(위의 예에서는 "abcde12345")를 제공해야 합니다.

```
[ec2-user ~]$ sudo openssl genrsa -aes128 -passout pass:abcde12345 -out custom.key 4096
```

- 예 4: 비 RSA 암호를 사용하여 키를 생성합니다. RSA 암호화는 두 개의 라지 소수의 결과를 기반으로 하는 공개 키의 크기 때문에 상대적으로 느릴 수 있습니다. 그러나 RSA 암호화 이외의 암호화를 사용하는 TLS의 키를 생성할 수 있습니다. 타원 곡선 수학을 기반으로 하는 키는 동등한 보안 수준을 제공할 때보다 작고 산술적으로 빠릅니다.

```
[ec2-user ~]$ sudo openssl ecparam -name prime256v1 -out custom.key -genkey
```

그 결과는 OpenSSL에서 지원하는 "명명된 곡선"인 prime256v1을 사용하는 256비트 타원 곡선 프라이빗 키입니다. [NIST](#)에 따르면 이 키의 암호화 강도는 2048비트 RSA 키보다 약간 더 높습니다.

Note

모든 CA에서 타원 곡선 기반 키에 대해 RSA 키와 동등한 수준의 지원을 제공하지는 않습니다.

새 프라이빗 키의 소유권 및 권한은 매우 제한적(소유자=루트, 그룹=루트, 소유자 전용 읽기/쓰기)이어야 합니다. 명령은 다음 예에서와 같습니다.

```
[ec2-user ~]$ sudo chown root:root custom.key
[ec2-user ~]$ sudo chmod 600 custom.key
```

```
[ec2-user ~]$ ls -al custom.key
```

이 명령의 결과는 다음과 같아야 합니다.

```
-rw----- root root custom.key
```

만족스러운 키를 생성 및 구성한 후 CSR을 생성할 수 있습니다.

- 원하는 키를 사용하여 CSR을 생성합니다. 다음 예에는 **custom.key**가 사용됩니다.

```
[ec2-user ~]$ sudo openssl req -new -key custom.key -out csr.pem
```

OpenSSL은 대화 상자를 열고 아래 표의 정보를 입력하라는 메시지를 표시합니다. 도메인에서 확인된 기본 호스트 인증서의 경우 Common Name을 제외한 모든 필드는 선택 사항입니다.

이름	설명	예제
국가 이름	해당 국가의 두 자리 ISO 약자.	US(=미국)
주 또는 지방 이름	해당 조직이 위치한 주 또는 지방의 이름. 이 이름은 약어로 사용할 수 없음.	워싱턴
시 이름	조직의 위치(예: 도시).	시애틀
조직 이름	해당 조직의 정식 이름. 조직 이름의 약칭을 사용하지 마세요.	Example Corporation
조직 단위 이름	조직에 대한 추가 정보(있는 경우).	부서 예
일반 이름	이 값은 사용자가 브라우저에 입력해야 하는 웹 주소와 정확히 일치해야 합니다. 일반적으로 이는 www.example.com 의 형식으로, 호스트 이름 또는 별칭이 앞에 붙는 도메인 이름을 뜻합니다. 자체 서명된 인증서로 DNS 확인 없이 테스트하는 경우, 일반 이름은 호스트 이름만으로 구성될 수 있습니다. CA는	www.example.com

이름	설명	예제
	*.example.com 과 같이 와일드 카드 이름을 허용하는 비싼 인증서도 제공합니다.	
이메일 주소	서버 관리자의 이메일 주소.	someone@example.com

마지막으로 OpenSSL은 챌린지 암호(선택 사항)를 입력하라는 메시지를 표시합니다. 이 암호는 해당 CSR 및 사용자와 해당 CA 간의 트랜잭션에만 적용되므로, 암호 및 기타 선택적 필드(선택적 회사 이름)에 대한 해당 CA의 권장 사항을 따릅니다. CSR 챌린지 암호는 서버 작업에 영향을 미치지 않습니다.

결과 파일인 **csr.pem**에는 퍼블릭 키, 퍼블릭 키의 디지털 서명 및 입력한 메타데이터가 포함되어 있습니다.

4. CA에 CSR을 제출합니다. 이는 보통 텍스트 편집기에서 CSR 파일을 열고 웹 양식에 내용을 복사하는 것으로 구성됩니다. 이때 인증서에 추가할 하나 이상의 주체 대체 이름(SAN)을 입력하라는 메시지가 나타날 수 있습니다. **www.example.com**이 일반 이름일 경우, **example.com**은 좋은 SAN이며, 그 반대의 경우도 마찬가지입니다. 사이트 방문자는 이 이름 중 하나를 입력하면 오류 없이 연결됩니다. CA 웹 양식에서 이를 허용하는 경우, SAN 목록에 일반 이름을 포함시킵니다. 일부 CA는 이를 자동으로 포함시킵니다.

요청이 승인되면 CA에서 서명한 새 호스트 인증서를 받게 됩니다. CA의 신뢰 체인을 완료하는 데 필요한 추가 인증서가 포함된 중간 인증서 파일을 다운로드하라는 안내를 받을 수도 있습니다.

Note

CA는 다양한 목적을 위해 마련된 여러 형식의 파일을 보낼 수 있습니다. 본 자습서에서는 PEM 형식의 인증서 파일만 사용해야 하는데, 이는 보통 .pem 또는 .crt 파일 확장명으로 표시되지만 항상 그런 것은 아닙니다. 어떤 파일을 사용할지 확실하지 않은 경우 텍스트 편집기로 파일을 열고 다음 라인으로 시작되는 블록 하나 이상이 포함되는 파일을 찾습니다.

```
- - - - -BEGIN CERTIFICATE - - - - -
```

또한 파일은 다음 라인으로 끝나야 합니다.

```
- - - - -END CERTIFICATE - - - - -
```

또한 명령줄의 파일을 다음과 같이 테스트할 수 있습니다.

```
[ec2-user certs]$ openssl x509 -in certificate.crt -text
```

이 줄이 파일에 나타나는지 확인하세요. .p7b, .p7c, 또는 유사한 파일 확장명으로 끝나는 파일을 사용하지 않습니다.

5. /etc/pki/tls/certs 디렉터리에 CA가 서명한 새 인증서와 모든 중간 인증서를 배치합니다.

Note

여러 가지 방법으로 새 인증서를 EC2 인스턴스에 업로드할 수 있지만, 가장 간편하고 유익한 방법은 텍스트 편집기(예: vi, nano, 메모장)를 로컬 컴퓨터와 인스턴스에 모두 열고 두 편집기 간에 파일 콘텐츠를 복사하여 붙이는 것입니다. EC2 인스턴스에서 이러한 작업을 수행할 때 루트 [sudo] 권한이 필요합니다. 이렇게 하면 권한 또는 경로 문제가 있는 경우 즉시 확인할 수 있습니다. 하지만 콘텐츠를 복사하는 동안 라인을 추가하거나 어떤 식으로든 콘텐츠를 변경하지 않도록 주의하세요.

/etc/pki/tls/certs 디렉터리 내에서 파일 소유권, 그룹 및 권한 설정이 매우 제한적인 AL2023 기본값(소유자=루트, 그룹=루트, 소유자 전용 읽기/쓰기)과 일치하는지 확인합니다. 다음 예제는 사용하는 명령을 보여 줍니다.

```
[ec2-user certs]$ sudo chown root:root custom.crt
[ec2-user certs]$ sudo chmod 600 custom.crt
[ec2-user certs]$ ls -al custom.crt
```

이 명령의 결과는 다음과 같아야 합니다.

```
-rw----- root root custom.crt
```

중간 인증서 파일에 대한 권한은 덜 엄격합니다(소유자=루트, 그룹=루트, 소유자 쓰기 가능, 그룹 읽기 가능, 모든 사용자 읽기 가능). 다음 예제는 사용하는 명령을 보여 줍니다.

```
[ec2-user certs]$ sudo chown root:root intermediate.crt
[ec2-user certs]$ sudo chmod 644 intermediate.crt
[ec2-user certs]$ ls -al intermediate.crt
```

이 명령의 결과는 다음과 같아야 합니다.

```
-rw-r--r-- root root intermediate.crt
```

6. /etc/pki/tls/private/ 디렉터리에서 CSR을 생성할 때 사용한 프라이빗 키를 배치합니다.

Note

여러 가지 방법으로 사용자 지정 키를 EC2 인스턴스에 업로드할 수 있지만, 가장 간편하고 유익한 방법은 텍스트 편집기(예: vi, nano, 메모장)를 로컬 컴퓨터와 인스턴스에 모두 열고 두 편집기 간에 파일 콘텐츠를 복사하여 붙이는 것입니다. EC2 인스턴스에서 이러한 작업을 수행할 때 루트 [sudo] 권한이 필요합니다. 이렇게 하면 권한 또는 경로 문제가 있는 경우 즉시 확인할 수 있습니다. 하지만 콘텐츠를 복사하는 동안 라인을 추가하거나 어떤 식으로든 콘텐츠를 변경하지 않도록 주의하세요.

/etc/pki/tls/private 디렉터리 내에서 파일 소유권, 그룹 및 권한 설정이 매우 제한적인 AL2023 기본값(소유자=루트, 그룹=루트, 소유자 전용 읽기/쓰기)과 일치하는지 다음 명령을 사용하여 확인합니다.

```
[ec2-user private]$ sudo chown root:root custom.key
[ec2-user private]$ sudo chmod 600 custom.key
[ec2-user private]$ ls -al custom.key
```

이 명령의 결과는 다음과 같아야 합니다.

```
-rw----- root root custom.key
```

7. 새 인증서 및 키 파일을 반영하기 위해 /etc/httpd/conf.d/ssl.conf를 편집합니다.

Note

SSLEngine on가 VirtualHost 블록에 설정되어 있는지 확인합니다.

- a. Apache의 SSLCertificateFile 명령에 CA가 서명한 호스트 인증서의 경로와 파일 이름을 입력합니다.

```
SSLCertificateFile /etc/pki/tls/certs/custom.crt
```

- b. 중간 인증서 파일을 받은 경우(이 예에서는 `intermediate.crt`), Apache의 `SSLCACertificateFile` 명령을 사용하여 경로 및 파일 이름을 입력합니다.

```
SSLCACertificateFile /etc/pki/tls/certs/intermediate.crt
```

Note

일부 CA는 호스트 인증서와 중간 인증서를 단일 파일로 결합하기 때문에 `SSLCACertificateFile` 명령이 불필요합니다. CA가 제공한 지침을 참조하세요.

- c. Apache의 `SSLCertificateKeyFile` 명령에 프라이빗 키(이 예에서는 `custom.key`)의 경로와 파일 이름을 입력합니다.

```
SSLCertificateKeyFile /etc/pki/tls/private/custom.key
```

8. `/etc/httpd/conf.d/ssl.conf`를 저장하고 Apache를 다시 시작합니다.

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

9. `https://` 접두사가 포함된 브라우저 URL 막대에 도메인 이름을 입력하여 서버를 테스트합니다. 브라우저에서는 테스트 페이지가 오류 생성 없이 HTTPS를 통해 로드되어야 합니다.

3단계: 보안 구성 테스트 및 하드닝

TLS이 작동되고 일반에 공개된 후 이의 실제 보안 수준을 테스트해야 합니다. 보안 설정을 무료로 완벽하게 분석해 주는 [Qualys SSL Labs](#)와 같은 온라인 서비스를 사용하면 이를 손쉽게 수행할 수 있습니다. 그 결과에 따라 수용할 프로토콜, 원하는 암호 및 제외할 암호를 관리하여 기본 보안 구성을 하드닝할 수 있습니다. 자세한 내용은 [how Qualys formulates its scores](#) 섹션을 참조하세요.

Important

실제 테스트는 서버 보안에 매우 중요합니다. 구성상의 작은 오류가 심각한 보안 침해 및 데이터 손실로 이어질 수 있습니다. 권장되는 보안 사례는 연구 및 새롭게 생겨나는 위협에 대처하기 위해 끊임없이 변화하므로 보안 감사를 주기적으로 실시하는 것이 서버 관리에 필수적입니다.

[Qualys SSL Labs](#) 사이트에 **www.example.com** 형식으로 서버의 정규화된 도메인 이름을 입력합니다. 약 2분 후 사이트 등급(A - F) 및 확인된 상세 분석 결과를 받게 됩니다. 아래 표에 AL2023 및 기본 Certbot 인증서의 기본 Apache 구성과 설정이 동일한 도메인에 대한 보고서가 요약되어 있습니다.

종합 등급	B
인증서	100%
프로토콜 지원	95%
키 교환	70%
암호화 수준	90%

개요에서 구성이 대체로 문제가 없어 보여도 세부 정보 보고서에서는 몇몇 잠재적 문제를 여기에 심각도 순서로 나열하여 표시합니다.

✗ RC4 암호는 이전 버전의 특정 브라우저에서 사용하도록 지원됩니다. 암호는 암호화 알고리즘의 수학적 핵심입니다. TLS 데이터 스트림을 암호화하는 데 사용하는 빠른 암호인 RC4에는 몇 가지 [심각한 취약점](#)이 있는 것으로 알려져 있습니다. 타당한 레거시 브라우저 지원 사유가 없다면 비활성화해야 합니다.

✗ 이전 TLS 버전이 지원됩니다. 구성에서는 TLS 1.0(이미 사용 중지 상태)과 TLS 1.1(사용 중지 절차 진행 중)을 지원합니다. 2018년부터는 TLS 1.2만 권장됩니다.

✗ 전방향 보안은 부분적으로 지원됩니다. [전방향 보안](#)은 프라이빗 키에서 파생된 임시(사용 후 삭제) 세션 키를 사용하여 암호화하는 알고리즘의 기능입니다. 이는 실제 공격자가 웹 서버의 장기 프라이빗 키를 보유하고 있더라도 HTTPS 데이터의 암호를 해독할 수 없다는 것을 뜻합니다.

TLS 구성을 수정하고 향후에 대비하려면

1. 텍스트 편집기에서 `/etc/httpd/conf.d/ssl.conf` 구성 파일을 열고 다음 줄의 시작 부분에 `"#"`을 입력하여 해당 줄을 주석으로 처리합니다.

```
#SSLProtocol all -SSLv3
```

2. 다음 명령을 추가합니다.

```
#SSLProtocol all -SSLv3
```



```
SSLProtocol -SSLv2 -SSLv3 -TLSv1 -TLSv1.1 +TLSv1.2
```

이러한 명령은 SSL 버전 2 및 3과 TLS 버전 1.0 및 1.1을 명시적으로 비활성화합니다. 이제 이 서버는 TLS 1.2 이외의 프로토콜을 사용하는 클라이언트와의 암호화된 연결을 허용하지 않습니다. 명령의 상세 내용은 서버의 구성 내용을 사람에게 더욱 명확히 전달합니다.

Note

이러한 방식으로 TLS 버전 1.0 및 1.1을 비활성화하면 적은 비율의 오래된 웹 브라우저가 사이트에 액세스하지 못하도록 차단합니다.

허용된 암호의 목록을 수정하려면

1. 구성 파일인 `/etc/httpd/conf.d/ssl.conf`에서 **SSLCipherSuite** 명령이 포함된 섹션을 찾고 기존의 줄을 줄의 시작에 “#”을 입력하여 주석으로 처리합니다.

```
#SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
```

2. 명시적 암호 그룹과 전방향 보안을 우선순위에 두고 부정확한 암호를 방지하는 암호 오더를 지정합니다. 여기에 사용된 **SSLCipherSuite** 명령은 서버에서 실행되는 특정 소프트웨어에 맞게 TLS 구성을 조정하는 [Mozilla SSL Configuration Generator](#)의 출력에 기반합니다. 먼저 다음 명령의 출력을 사용하여 Apache와 OpenSSL의 버전을 확인합니다.

```
[ec2-user ~]$ yum list installed | grep httpd
```

```
[ec2-user ~]$ yum list installed | grep openssl
```

예를 들어, 반환된 정보가 Apache 2.4.34 및 OpenSSL 1.0.2인 경우 이를 생성기에 입력합니다. "현대" 호환성 모델을 선택하면 적극적으로 보안을 적용하지만 대부분의 브라우저에서 여전히 작동하는 **SSLCipherSuite** 명령을 생성합니다. 소프트웨어가 최신 구성을 지원하지 않으면 소프트웨어를 업데이트하거나 대신 "중간" 구성을 선택할 수 있습니다.

```
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
```

선택된 암호에는 이름에 Elliptic Curve Diffie-Hellman Ephemeral의 약자인 ECDHE가 포함되어 있습니다. ephemeral은 전방향 보안을 나타냅니다. 부차적 결과로서 해당 암호는 RC4를 지원하지 않습니다.

내용이 표시되지 않는 기본값 또는 terse 명령 대신 명시적 암호 목록을 사용하는 것이 좋습니다.

생성된 명령을 `/etc/httpd/conf.d/ssl.conf`에 복사합니다.

Note

여기에서는 가독성을 위해 여러 줄로 표시했지만, 이 명령은 `/etc/httpd/conf.d/ssl.conf`에 복사할 때 암호 이름 사이에 공백 없이 콜론만을 추가하여 한 줄에 입력해야 합니다.

- 마지막으로 줄 시작 부분에 있는 `"#"`을 제거하여 다음 줄의 주석 처리를 해제합니다.

```
#SSLHonorCipherOrder on
```

이 명령은 (이 예에서는) 전방향 보안을 지원하는 암호를 포함하여 서버에서 순위가 높은 암호를 선호하도록 합니다. 이 명령이 설정되면 서버는 먼저 강력한 보안 연결 설정을 시도해 본 후 보안이 더 약한 허용된 암호로 대체합니다.

이 두 절차를 모두 완료한 다음에는 변경 사항을 `/etc/httpd/conf.d/ssl.conf`에 저장하고 Apache를 재시작합니다.

[Qualys SSL Labs](#)에서 도메인을 다시 테스트하면 RC4 취약성과 다른 경고 문제가 해결되고 요약은 다음과 같을 것입니다.

종합 등급	A
인증서	100%
프로토콜 지원	100%
키 교환	90%
암호화 수준	90%

OpenSSL을 업데이트할 때마다 새 암호가 사용되고 이전 암호에 대한 지원은 제거됩니다. EC2 AL2023 인스턴스를 최신으로 유지하고, [OpenSSL](#)의 보안 알리를 잘 확인하며, 기술 정보 신문의 새 보안 취약점에 대한 보고를 잘 살핍니다.

문제 해결

- 암호를 입력하지 않으면 Apache 웹 서버가 시작되지 않음

암호화되고 암호로 보호되는 프라이빗 서버 키를 설치한 경우 이는 예상된 동작입니다.

키에서 암호화 및 암호 요구 사항을 제거할 수 있습니다. 기본 디렉터리에 `custom.key`라는 암호화된 프라이빗 RSA 키가 있고 이 키의 암호가 **abcde12345**라고 가정하면, EC2 인스턴스에서 다음 명령을 실행하여 이 키의 암호화되지 않은 버전을 생성합니다.

```
[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
```

이제 Apache가 암호를 묻지 않고 시작할 것입니다.

- `sudo dnf install -y mod_ssl`을 실행할 때 오류가 발생합니다.

SSL에 필요한 패키지를 설치하려 할 때 다음과 같은 오류가 표시될 수 있습니다.

```
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
```

이 오류는 일반적으로 EC2 인스턴스에서 AL2023이 실행되고 있지 않음을 의미합니다. 이 자습서는 공식 AL2023 AMI에서 새로 생성된 인스턴스만 지원합니다.

자습서: AL2023에서 WordPress 블로그 호스트

다음 절차는 AL2023 인스턴스에서 WordPress 블로그를 설치, 구성 및 보안 설정하는 데 도움이 됩니다. 본 자습서는 기존 호스팅 서비스에서는 일반적이지 않은 WordPress 블로그를 호스팅하는 웹 서버를 사용자가 완전히 제어할 수 있다는 점에서 Amazon EC2 사용에 있어 좋은 입문 기회를 제공합니다.

사용자는 서버에 대한 소프트웨어 패키지를 업데이트하고 보안 패치를 유지관리할 책임이 있습니다. 웹 서버 구성과 직접 상호 작용할 필요가 없는 보다 자동화된 WordPress 설치를 위해 CloudFormation 서비스는 빠르게 시작할 수 있는 WordPress 템플릿을 제공합니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 [시작하기](#)를 참조하세요. 데이터베이스가 분리된 고가용성 솔루션이 필요하다면 AWS Elastic Beanstalk 개발자 안내서에서 [고가용성 WordPress 웹 사이트 배포](#)를 참조하세요.

Important

이 절차는 AL2023에서 사용하기 위한 것입니다. 기타 배포에 대한 자세한 내용은 해당 설명서를 참조하세요. 본 자습서에 있는 단계의 상당수가 Ubuntu 인스턴스에서 작동하지 않습니다. Ubuntu 인스턴스에 WordPress를 설치하는 방법은 Ubuntu 설명서에서 [WordPress](#) 섹션을 참조하세요. [CodeDeploy](#)를 사용하여 Amazon Linux, macOS 또는 Unix 시스템에서 태스크를 수행할 수도 있습니다.

주제

- [사전 조건](#)
- [WordPress 설치](#)
- [다음 단계](#)
- [도움말! 내 퍼블릭 DNS 이름이 변경되어 블로그를 사용할 수 없습니다.](#)

사전 조건

탄력적 IP 주소(EIP)는 WordPress 블로그를 호스팅하는 데 사용 중인 인스턴스와 연결하는 것이 가장 바람직합니다. 인스턴스의 퍼블릭 DNS 주소가 설치 위치를 바꾸거나 위반하는 것을 방지할 수 있기 때문입니다. 자신이 소유하고 있는 도메인 이름을 블로그에 사용하고 싶다면 도메인 이름의 DNS 레코드가 EIP 주소를 가리키도록 업데이트할 수 있습니다(이와 관련하여 도움이 필요하다면 도메인 이름 등록 기관에게 문의하세요). 실행 중인 인스턴스와 연결되어 있는 EIP 주소는 한 개까지 무료로 사용할 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서에서 [탄력적 IP 주소](#)를 참조하세요. [자습서: AL2023에 LAMP 서버 설치](#) 자습서는 보안 그룹이 HTTP 및 HTTPS 트래픽을 허용하도록 구성하는 단계와 파일 권한이 웹 서버에 맞게 적절하게 설정되어 있는지 확인하는 여러 단계를 포함하고 있습니다. 보안 그룹 규칙 추가에 대한 자세한 내용은 [보안 그룹에 규칙 추가](#) 섹션을 참조하세요.

블로그에 사용할 도메인 이름이 아직 없는 경우 Route 53에 도메인 이름을 등록하고 인스턴스의 EIP 주소를 도메인 이름에 연결할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서의 [Amazon Route 53을 사용하여 도메인 이름 등록](#)을 참조하세요.

WordPress 설치

인스턴스에 연결한 후 WordPress 설치 패키지를 다운로드합니다. 인스턴스 연결에 대한 자세한 내용은 [AL2023 인스턴스에 연결](#) 섹션을 참조하세요.

1. 다음 명령을 사용하여 이러한 패키지를 다운로드하고 설치합니다.

```
dnf install wget php-mysqlnd httpd php-fpm php-mysqli mariadb105-server php-json
php php-devel -y
```

2. 출력에 다음과 비슷한 문구의 경고가 표시될 수 있습니다(버전은 시간에 따라 다를 수 있음).

```
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
```

OS를 가능한 최신 상태로 유지하는 것이 좋지만 환경에 충돌이 없는지 확인하기 위해 각 버전을 반복할 수 있습니다. 1단계에서 설명한 이전 패키지 설치에 실패할 경우 나열된 최신 릴리스 중 하나로 업데이트하고 다시 시도해야 할 수 있습니다.

3. wget 명령을 사용하여 최신 WordPress 설치 패키지를 다운로드 합니다. 다음 명령을 사용할 경우 언제나 최신 릴리스를 다운로드합니다.

```
[ec2-user ~]$ wget https://wordpress.org/latest.tar.gz
```

4. 설치 패키지의 압축 및 아카이빙을 해제합니다. 설치 폴더는 wordpress라는 폴더로 압축 해제됩니다.

```
[ec2-user ~]$ tar -xzf latest.tar.gz
```

WordPress 설치에 대한 데이터베이스 사용자 및 데이터베이스를 생성하려면

WordPress 설치 시 블로그 게시물, 사용자 의견 등의 정보를 데이터베이스에 저장해야 합니다. 다음 절차를 통해 블로그의 데이터베이스와 이 데이터베이스에 대해 정보 읽기 및 저장 권한이 있는 사용자를 생성할 수 있습니다.

1. 데이터베이스와 웹 서버를 시작합니다.

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

2. 데이터베이스 서버에 root 사용자로 로그인합니다. 메시지가 표시되면 데이터베이스 root 암호를 입력합니다. 이 암호는 사용자의 root 시스템 암호와 다를 수 있으며, 데이터베이스 서버를 보안 설정하지 않은 경우 암호가 비어 있을 수도 있습니다.

데이터베이스 서버를 보안 설정하지 않았다면 반드시 설정하시기 바랍니다. 자세한 내용은 [3단계: 데이터베이스 서버 보안 설정\(AL2023\)](#) 섹션을 참조하세요.

```
[ec2-user ~]$ mysql -u root -p
```

3. MySQL 데이터베이스에 대한 사용자 및 암호를 생성합니다. WordPress 설치에 MySQL 데이터베이스를 통신하기 위해 상기 값을 사용합니다. 고유한 사용자 이름과 암호로 해당 부분을 대체하여 다음 명령을 입력합니다.

```
CREATE USER 'wordpress-user'@'localhost' IDENTIFIED BY 'your_strong_password';
```

사용자에 대해 보안이 강력한 암호를 생성하시기 바랍니다. 작은따옴표(')는 각 명령을 구별하는 구분자로 기능하기 때문에, 암호에는 사용하지 마세요. 기존 암호를 재사용하지 마세요. 새로 설정한 암호는 안전한 장소에 보관하세요.

4. 데이터베이스를 생성합니다. 데이터베이스에 이를 설명할 수 있는 유의미한 이름을 붙입니다(예: wordpress-db.).

Note

아래 명령에서 데이터베이스 이름을 앞 뒤로 묶는 기호(`)를 백틱(backtick)이라고 합니다. 백틱(`) 키는 일반적으로 표준 키보드에서 Tab 키 위에 있습니다. 백틱이 항상 필요하지는 않지만, 이를 통해 데이터베이스 이름에 하이픈(-) 등 허용되지 않는 문자를 사용할 수 있습니다.

```
CREATE DATABASE `wordpress-db`;
```

5. 데이터베이스에 대한 전체 권한을 이전에 생성한 WordPress 사용자에게 부여합니다.

```
GRANT ALL PRIVILEGES ON `wordpress-db`.* TO "wordpress-user"@"localhost";
```

6. 데이터베이스 권한을 새로고침(flush)해서 모든 변경사항이 적용되도록 합니다.

```
FLUSH PRIVILEGES;
```

7. mysql 클라이언트를 종료합니다.

```
exit
```

wp-config.php 파일 생성 및 편집 방법

WordPress 설치 폴더는 wp-config-sample.php라는 샘플 구성 파일을 포함하고 있습니다. 본 절차에서는 이 파일을 복사하고 특정 구성에 맞도록 편집합니다.

1. wp-config-sample.php 파일을 wp-config.php라는 파일에 복사합니다. 이를 통해 새 구성 파일을 생성하고 원본 샘플 파일을 이전 상태 그대로 백업으로 보존할 수 있습니다.

```
[ec2-user ~]$ cp wordpress/wp-config-sample.php wordpress/wp-config.php
```

2. wp-config.php 파일을 원하는 텍스트 편집기(nano, vim 등)로 편집하고 설치에 대한 값을 입력합니다. 원하는 텍스트 편집기가 없는 경우 초보자에게는 nano가 적합합니다.

```
[ec2-user ~]$ nano wordpress/wp-config.php
```

- a. DB_NAME을(를) 정의하는 줄을 찾고 database_name_here을(를) [Step 4의 WordPress 설치에 대한 데이터베이스 사용자 및 데이터베이스를 생성하려면](#)에서 생성한 데이터베이스 이름으로 변경합니다.

```
define('DB_NAME', 'wordpress-db');
```

- b. DB_USER을(를) 정의하는 줄을 찾고 username_here을(를) [Step 3의 WordPress 설치에 대한 데이터베이스 사용자 및 데이터베이스를 생성하려면](#)에서 생성한 데이터베이스 사용자로 변경합니다.

```
define('DB_USER', 'wordpress-user');
```

- c. DB_PASSWORD을(를) 정의하는 줄을 찾고 password_here을(를) [Step 3의 WordPress 설치에 대한 데이터베이스 사용자 및 데이터베이스를 생성하려면](#)에서 생성한 보안성이 강력한 암호로 변경합니다.

```
define('DB_PASSWORD', 'your_strong_password');
```

- d. Authentication Unique Keys and Salts라는 섹션을 검색합니다. 이 KEY 및 SALT 값은 WordPress 사용자가 로컬 컴퓨터에 저장하는 브라우저 쿠키에 암호 계층을 제공합니다. 기본적으로 긴 무작위 값을 추가해서 사이트의 보안성을 강화할 수 있습니다. <https://api.wordpress.org/secret-key/1.1/salt/>을 방문해서 키 값의 세트를 무작위로 생성하고 이를 wp-config.php 파일로 복사해서 붙여 넣을 수 있습니다. PuTTY 터미널로 텍스트를 붙여넣기 하기 위해, PuTTY 터미널 내부에서 텍스트를 붙여넣기하려는 위치에 커서를 놓고 마우스를 오른쪽 클릭합니다.

보안 키에 대한 자세한 내용을 보려면 http://codex.wordpress.org/Editing_wp-config.php#Security_Keys로 이동하세요.

Note

아래 값은 예시 목적만을 위한 것입니다. 설치 시 이 값을 사용하지는 마세요.

```
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. 파일을 저장하고 텍스트 편집기를 종료합니다.

WordPress 파일을 Apache 문서 루트 아래에 설치하려면

- 설치 폴더 압축을 해제하고 MySQL 데이터베이스 및 맞춤형 WordPress 구성 파일을 사용자 설정했으므로, 이제 설치 파일을 웹 서버 문서 루트에 복사하여 설치를 완료하는 설치 스크립트를 실행할 수 있습니다. 이 파일의 위치는 WordPress 블로그를 웹 서버의 실제 루트(예: *my.public.dns.amazonaws.com*)에서 사용하도록 할지 아니면 루트 아래의 하위 디렉터리나 폴더(예: *my.public.dns.amazonaws.com/blog*)에서 사용하도록 할지에 따라 다릅니다.
- WordPress를 문서 루트에서 실행하려면 WordPress 설치 디렉터리의 파일(디렉터리 자체는 제외)을 다음과 같이 복사합니다.

```
[ec2-user ~]$ cp -r wordpress/* /var/www/html/
```

- WordPress를 문서 루트의 대체 디렉터리에서 실행하려면 먼저 해당 디렉터리를 생성한 후 파일을 그 디렉터리로 복사합니다. 이 예에서는 다음과 같이 WordPress가 blog 디렉터리에서 실행됩니다.

```
[ec2-user ~]$ mkdir /var/www/html/blog
[ec2-user ~]$ cp -r wordpress/* /var/www/html/blog/
```

Important

다음 프로시저로 즉시 이동하지 않는 경우는 보안상 문제가 발생할 수 있으므로 Apache 웹 서버(httpd)를 중단하세요. Wordpress 설치를 Apache 문서 루트 아래로 이동한 후에는 WordPress 설치 스크립트가 보호되지 않는 상태이기 때문에 Apache 웹 서버가 실행 중일 때 블로그에 침입자가 액세스할 가능성이 있습니다. Apache 웹 서버를 중지하려면 `sudo service httpd stop` 명령을 입력합니다. 다음 절차로 즉시 이동하는 경우는 Apache 웹 서버를 중단시킬 필요가 없습니다.

WordPress에서 퍼머링크(permalinks)를 사용하는 방법

WordPress가 올바르게 작동하려면 Apache `.htaccess` 파일을 사용해야 하지만 Amazon Linux에서는 기본적으로 이 파일을 사용할 수 없습니다. 따라서 아래 방법에 따라 Apache 문서 루트에서 모든 재정의의 허용해야 합니다.

1. 자주 사용하는 텍스트 편집기(`httpd.conf` 또는 `nano`)로 `vim` 파일을 엽니다. 원하는 텍스트 편집기가 없는 경우 초보자에게는 `nano`가 적합합니다.

```
[ec2-user ~]$ sudo vim /etc/httpd/conf/httpd.conf
```

2. 다음과 같이 시작하는 영역을 찾습니다. `<Directory "/var/www/html">`

```
<Directory "/var/www/html">
#
# Possible values for the Options directive are "None", "All",
# or any combination of:
#   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
#
# Note that "MultiViews" must be named *explicitly* --- "Options All"
# doesn't give it to you.
#
# The Options directive is both complicated and important. Please see
# http://httpd.apache.org/docs/2.4/mod/core.html#options
# for more information.
#
Options Indexes FollowSymLinks

#
# AllowOverride controls what directives may be placed in .htaccess files.
# It can be "All", "None", or any combination of the keywords:
#   Options FileInfo AuthConfig Limit
#
AllowOverride None

#
# Controls who can get stuff from this server.
#
Require all granted
</Directory>
```

3. 위 영역에서 `AllowOverride None` 라인을 `AllowOverride All`로 변경합니다.

Note

이 파일에는 AllowOverride 라인이 많기 때문에 <Directory "/var/www/html"> 영역의 라인을 변경할 때는 주의해야 합니다.

AllowOverride **ALL**

4. 파일을 저장하고 텍스트 편집기를 종료합니다.

AL2023에서 PHP 그래픽 그리기 라이브러리를 설치하는 방법

PHP용 GD 라이브러리를 사용하면 이미지를 수정할 수 있습니다. 블로그의 헤더 이미지를 잘라야 하는 경우 이 라이브러리를 설치합니다. 설치하는 phpMyAdmin 버전에는 이 라이브러리의 특정 최소 버전(예: 버전 8.1)이 필요할 수 있습니다.

AL2023에서 PHP 그래픽 그리기 라이브러리를 설치하려면 다음 명령을 사용합니다. 예를 들어, LAMP 스택 설치의 일부로 소스에서 php8.1을 설치한 경우 이 명령은 PHP 그래픽 그리기 라이브러리의 버전 8.1을 설치합니다.

```
[ec2-user ~]$ sudo dnf install php-gd
```

설치된 버전을 확인하려면 다음 명령을 사용합니다.

```
[ec2-user ~]$ sudo dnf list installed | grep php-gd
```

다음은 예제 출력입니다.

php-gd.x86_64	8.1.30-1.amzn2	@amazonlinux

Amazon Linux AMI에서 PHP 그래픽 그리기 라이브러리를 설치하려면

PHP용 GD 라이브러리를 사용하면 이미지를 수정할 수 있습니다. 블로그의 헤더 이미지를 잘라야 하는 경우 이 라이브러리를 설치합니다. 설치하는 phpMyAdmin 버전에는 이 라이브러리의 특정 최소 버전(예: 버전 8.1)이 필요할 수 있습니다.

사용 가능한 버전을 확인하려면 다음 명령을 사용합니다.

```
[ec2-user ~]$ dnf list | grep php
```

다음은 PHP 그래픽 그리기 라이브러리(버전 8.1) 출력의 예입니다.

```
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
```

Amazon Linux AMI에서 특정 버전의 PHP 그래픽 그리기 라이브러리(예: 버전 php8.1)를 설치하려면 다음 명령을 사용합니다.

```
[ec2-user ~]$ sudo dnf install -y php8.1-gd
```

Apache 웹 서버에 대한 파일 권한 수정 방법

WordPress의 제공 기능 중 일부(예: 관리 화면을 통한 미디어 업로드 등)는 Apache 문서 루트에 대한 쓰기 권한을 필요로 합니다. 이미 적용하지 않았다면 다음의 그룹 멤버십 및 권한([LAMP 웹 서버 자습서](#)에서 자세히 설명)을 적용합니다.

1. /var/www의 파일 소유권 및 그 콘텐츠를 apache 사용자에게 허용합니다.

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

2. /var/www 및 그 콘텐츠의 그룹 소유권을 apache 그룹에 허용합니다.

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

3. /var/www 및 그 하위 디렉터리의 디렉터리 권한을 변경해서 그룹 쓰기 권한을 추가하고 미래 하위 디렉터리에서 그룹 ID를 설정합니다.

```
[ec2-user ~]$ sudo chmod 2775 /var/www
```

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

4. /var/www 및 그 하위 디렉터리의 파일 권한을 재귀적으로 변경합니다.

```
[ec2-user ~]$ find /var/www -type f -exec sudo chmod 0644 {} \;
```

Note

WordPress를 FTP 서버로도 사용하려는 경우 여기서 더 많은 권한 그룹 설정이 필요합니다. 이 작업을 수행하려면 권장된 [WordPress 단계 및 보안 설정](#)을 검토하세요.

5. Apache 웹 서버를 재시작해서 새 그룹 및 권한을 가져옵니다.

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

AL2023으로 WordPress 설치 스크립트를 실행하는 방법

이제 WordPress를 설치할 준비가 되었습니다. 사용하는 명령은 운영 체제에 따라 다릅니다. 이 절차의 명령은 AL2023에서 사용하기 위한 것입니다. 다음의 절차를 AL2023 AMI와 함께 사용하세요.

1. systemctl 명령을 사용하여 시스템이 부팅될 때마다 httpd 및 데이터베이스 서비스가 시작되도록 합니다.

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

2. 데이터베이스 서버가 실행되는지 확인합니다.

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

데이터베이스 서비스가 실행 중이지 않은 경우, 이를 시작합니다.

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

3. Apache 웹 서버(httpd)가 실행 중인지 확인합니다.

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

httpd 서비스가 실행 중이지 않은 경우, 이를 시작합니다.

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

4. 웹 브라우저에서 WordPress 블로그의 URL을 입력합니다(인스턴스에 대한 퍼블릭 DNS 주소 또는 blog 폴더 다음의 주소). 이제 WordPress 설치 스크립트가 나타납니다. WordPress 설치에 필요한 정보를 제공합니다. WordPress 설치(Install WordPress)를 선택해서 설치를 완료합니다. 자세한 내용은 WordPress 웹 사이트의 [5단계: 설치 스크립트 실행](#)을 참조하세요.

AL2023 AMI로 WordPress 설치 스크립트를 실행하는 방법

1. chkconfig 명령을 사용하여 시스템이 부팅될 때마다 httpd 및 데이터베이스 서비스가 시작되도록 합니다.

```
[ec2-user ~]$ sudo chkconfig httpd on && sudo chkconfig mariadb on
```

2. 데이터베이스 서버가 실행되는지 확인합니다.

```
[ec2-user ~]$ sudo service mariadb status
```

데이터베이스 서비스가 실행 중이지 않은 경우, 이를 시작합니다.

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

3. Apache 웹 서버(httpd)가 실행 중인지 확인합니다.

```
[ec2-user ~]$ sudo service httpd status
```

httpd 서비스가 실행 중이지 않은 경우, 이를 시작합니다.

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

4. 웹 브라우저에서 WordPress 블로그의 URL을 입력합니다(인스턴스에 대한 퍼블릭 DNS 주소 또는 blog 폴더 다음의 주소). 이제 WordPress 설치 스크립트가 나타납니다. WordPress 설치에 필요한 정보를 제공합니다. WordPress 설치(Install WordPress)를 선택해서 설치를 완료합니다. 자세한 내용은 WordPress 웹 사이트의 [5단계: 설치 스크립트 실행](#)을 참조하세요.

다음 단계

WordPress 블로그를 테스트한 후 구성을 업데이트하세요.

사용자 지정 도메인 이름 사용

EC2 인스턴스의 EIP 주소와 연결되어 있는 도메인 이름이 있는 경우에는 EC2 퍼블릭 DNS 주소 대신에 해당 이름을 사용하여 블로그를 구성할 수 있습니다. 자세한 내용은 WordPress 웹 사이트의 [사이트 URL 변경](#)을 참조하세요.

블로그 구성

다른 [테마](#)와 [플러그인](#)을 사용하여 더욱 풍부한 맞춤형 경험을 독자에게 제공하도록 블로그를 구성할 수도 있습니다. 하지만 설치 프로세스가 역효과를 낳아 전체 블로그를 잃는 경우가 발생할 수도 있습니다. 따라서 테마나 플러그인을 설치하기 전에 인스턴스의 백업 Amazon Machine Image(AMI)를 생성하여 설치 중 오류가 발생하더라도 블로그를 복구할 수 있도록 대비하는 것이 좋습니다. 자세한 내용은 Amazon EC2 사용 설명서의 [고유의 AMI 생성](#) 섹션을 참조하세요.

용량 증가

운영하는 WordPress 블로그가 유명해지고 그에 따라 보다 많은 컴퓨팅 파워 또는 스토리지가 필요하게 될 경우 다음 단계를 고려하세요.

- 인스턴스에서 스토리지 공간을 확장합니다. 자세한 내용은 [Amazon EBS Elastic Volumes](#) 섹션을 참조하세요.
- MySQL 데이터베이스를 [Amazon RDS](#)로 이동하여 이 서비스의 간편한 조정 기능을 이용합니다.

인터넷 트래픽의 네트워크 성능 향상

블로그가 전 세계에 위치한 사용자로부터 트래픽을 유도할 것으로 예상되는 경우 [AWS Global Accelerator](#) 사용을 고려해 보세요. Global Accelerator는 사용자의 클라이언트 디바이스와 AWS에서 실행되는 WordPress 애플리케이션 간의 인터넷 트래픽 성능을 개선하여 지연 시간을 줄일 수 있도록 돕습니다. Global Accelerator는 [AWS 글로벌 네트워크](#)를 사용하여 클라이언트와 가장 가까운 AWS 리전의 정상 애플리케이션 엔드포인트로 트래픽을 전달합니다.

WordPress에 대해 자세히 알아보기

다음 링크에는 WordPress에 대한 자세한 정보가 포함되어 있습니다.

- WordPress에 대한 자세한 내용은 [Codex](#)에서 WordPress Codex 도움 설명서를 참조하세요.

- 설치 문제 해결에 대한 자세한 내용은 [일반적인 설치 문제](#) 섹션을 참조하세요.
- WordPress 블로그 보안 강화에 대한 자세한 내용은 [WordPress 하드닝](#) 섹션을 참조하세요.
- WordPress 블로그를 최신 상태로 유지하는 방법에 대한 자세한 내용은 [WordPress 업데이트](#) 섹션을 참조하세요.

도움말! 내 퍼블릭 DNS 이름이 변경되어 블로그를 사용할 수 없습니다.

WordPress 설치 위치는 EC2 인스턴스의 퍼블릭 DNS 주소를 사용해 자동으로 구성됩니다. 이때 인스턴스를 중단했다가 다시 시작하면 퍼블릭 DNS 주소가 바뀌어(탄력적 IP 주소와 연결되어 있지 않은 경우) 블로그를 더 이상 사용할 수 없게 됩니다. 리소스를 참조해야 할 주소가 더 이상 존재하지 않거나 다른 EC2 인스턴스에 할당되었기 때문입니다. 이 문제를 비롯해 몇 가지 해결책에 대한 자세한 내용은 <https://wordpress.org/support/article/changing-the-site-url/>을 참조하세요.

이 문제가 WordPress 설치 위치에 발생하더라도 아래 절차에 따라 WordPress의 wp-cli 명령줄 인터페이스를 사용하면 블로그를 복구할 수 있습니다.

wp-cli를 사용하여 WordPress 사이트 URL을 바꾸는 방법

1. SSH를 통해 EC2 인스턴스에 연결합니다.
2. 인스턴스의 이전 사이트 URL과 새로운 사이트 URL을 기록합니다. 이전 사이트 URL은 WordPress 설치 시 EC2 인스턴스의 퍼블릭 DNS 이름일 가능성이 높습니다. 그리고 새로운 사이트 URL은 EC2 인스턴스의 현재 퍼블릭 DNS 이름입니다. 이전 사이트 URL을 잘 모르더라도 아래와 같이 curl 명령을 사용하여 찾을 수 있습니다.

```
[ec2-user ~]$ curl localhost | grep wp-content
```

명령을 실행하여 출력되는 화면에서 이전 퍼블릭 DNS 이름의 참조를 확인해야 합니다. 출력 화면은 다음과 같습니다(빨간색의 이전 사이트 URL).

```
<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. 다음 명령으로 wp-cli를 다운로드합니다.

```
[ec2-user ~]$ curl -O https://raw.githubusercontent.com/wp-cli/builds/gh-pages/phar/wp-cli.phar
```


- 아래와 같은 명령으로 이전 사이트 URL을 찾아 WordPress 설치 위치로 바꿉니다. EC2 인스턴스의 이전 사이트 URL과 새로운 사이트 URL, 그리고 WordPress 설치 경로(일반적으로 /var/www/html 또는 /var/www/html/blog)를 치환합니다.

```
[ec2-user ~]$ php wp-cli.phar search-replace 'old_site_url' 'new_site_url' --path=/path/to/wordpress/installation --skip-columns=guid
```

- 웹 브라우저에서 WordPress 블로그의 새로운 사이트 URL을 입력하여 사이트에 올바르게 접속되는지 다시 확인합니다. 그렇지 않은 경우 [사이트 URL 변경](#) 및 [일반적인 설치 문제](#) 섹션에서 자세한 내용을 참조하세요.

자습서: Redis 6에서 AL2023의 Valkey로 전환

다음 설명서에서는 AL2023의 Redis 6에서 Valkey로 전환하는 주요 측면을 설명합니다.

Redis 6 지원 타임라인

Redis 6는 2027년 1월 31일에 수명 종료(EOL)에 도달했습니다. 이 날짜부터 Redis 프로젝트로부터 Redis 6는 더 이상 업데이트 또는 보안 패치를 받지 않습니다. 지속적인 지원 및 보안 업데이트를 보장하려면 2027년 1월 이전에 Valkey로 마이그레이션하는 것이 좋습니다.

Redis 버전 지원 타임라인에 대한 자세한 내용은 [Redis End-Of-Life 일정](#) 설명서를 참조하세요.

Valkey 소개

Valkey는 Linux Foundation에서 유지 관리하는 Redis 7의 오픈 소스 포크입니다. Redis OSS(Open Source Software) 버전 2.x~7.2.x와 완벽하게 호환됩니다. Valkey는 익숙한 Redis API 및 기능을 유지하면서 몇 가지 향상된 기능을 제공합니다.

- 멀티스레딩을 통해 성능이 향상.
- 특히 클러스터 모드에서 메모리 효율성이 향상.
- 데이터 일관성을 높이기 위한 이중 채널 복제.

마이그레이션 계획 및 타임라인

Redis 6가 수명 종료(EOL)에 도달하면 사용자는 2027년 1월 31일 이전에 Redis 6에서 Valkey로 마이그레이션하는 것이 좋습니다. 이 마이그레이션은 수동 개입이 필요하며 자동이 아닙니다.

Amazon Linux는 Redis 종속 애플리케이션에 대한 지속적인 기능, 지원 및 보안 업데이트를 보장하기 위해 이 마이그레이션을 권장합니다.

마이그레이션 옵션 및 단계

배포 요구 사항 및 운영 요구 사항에 따라 Valkey에 대한 세 가지 마이그레이션 경로를 제안합니다.

옵션 1: 새 인스턴스 설치

새 배포의 경우 또는 데이터 마이그레이션이 필요하지 않은 경우:

1. Valkey 설치:

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

2. Valkey 시작:

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

3. (선택 사항) 부팅 시 Valkey 활성화:

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

4. 설치를 확인합니다.

```
[ec2-user ~]$ valkey-cli info server  
[ec2-user ~]$ valkey-cli ping
```

옵션 2: 인플레이스 교체

데이터 지속성이 필요하지 않은 기존 인스턴스의 경우:

1. Redis 6 중지:

```
[ec2-user ~]$ sudo systemctl stop redis6
```

2. Valkey 설치:

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

3. (선택 사항) Valkey에서 Redis 6 구성 사용:

```
[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. (선택 사항) Valkey에서 Redis 6 감시 구성 파일 사용:

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

5. Valkey 시작:

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

6. (선택 사항) 부팅 시 Valkey 활성화:

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

7. Valkey 설치 확인:

```
[ec2-user ~]$ valkey-cli info server
[ec2-user ~]$ valkey-cli ping
```

8. Redis 6 제거:

```
[ec2-user ~]$ sudo dnf remove redis6
```

옵션 3: 데이터 마이그레이션

이 옵션을 사용하면 Redis 6과 Valkey를 동시에 실행할 수 있습니다.

1. Redis 6을 제거하지 않고 Valkey를 설치합니다.

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

2. (선택 사항) Valkey에서 Redis 6 구성 사용:

```
[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. (선택 사항) Valkey에서 Redis 6 감시 구성 파일 사용:

```
[ec2-user ~]$ sudo cp /etc/redis6/sentinel.conf /etc/valkey/sentinel.conf
[ec2-user ~]$ sudo chown valkey:root /etc/valkey/sentinel.conf
```

4. Valkey 구성 수정:

/etc/valkey/valkey.conf를 편집하고 Redis 6과의 충돌을 방지하기 위해 'port' 지시문을 다른 값(예: 6380)으로 설정합니다.

5. Valkey 시작:

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

6. (선택 사항) 부팅 시 Valkey 활성화:

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

7. Valkey 설치 확인:

```
[ec2-user ~]$ valkey-cli -p port info server
[ec2-user ~]$ valkey-cli -p port ping
```

Note

##를 구성된 포트 번호로 바꿉니다.

8. 데이터 마이그레이션:

이제 복제 또는 수동 데이터 전송 방법을 사용하여 Redis 6에서 Valkey로 데이터를 마이그레이션할 수 있습니다.

9. 애플리케이션 구성 업데이트:

Valkey 포트를 사용하도록 애플리케이션을 점진적으로 업데이트합니다.

10. Redis 6 제거:

모든 데이터와 애플리케이션이 마이그레이션되면 Redis 6를 중지하고 제거할 수 있습니다.

```
[ec2-user ~]$ sudo systemctl stop redis6  
[ec2-user ~]$ sudo dnf remove redis6
```

Note

프로덕션 시스템에서 변경 사항을 구현하기 전에 테스트 환경에서 마이그레이션 프로세스를 검증할 것을 강력히 권장합니다.

관련 주제

Valkey에 대한 자세한 내용은 다음을 참조하세요.

- Valkey: <https://valkey.io/>
- Valkey 마이그레이션: <https://valkey.io/topics/migration/>

자습서: AL2023에 GNOME 데스크톱 환경 설치

[GNOME 데스크톱 환경](#)은 릴리스 2023.7 이상부터 AL2023용 그래픽 사용자 인터페이스 옵션으로 사용할 수 있습니다.

다음 절차는 AL2023 인스턴스에 GNOME 데스크톱 환경을 설치하는 데 도움이 됩니다. 이 그래픽 인터페이스를 사용하면 명령줄 인터페이스만 사용하는 대신 익숙한 데스크톱 환경을 사용하여 Linux 시스템과 상호 작용할 수 있습니다.

내용

- [사전 조건](#)
- [설치](#)
- [관련 주제](#)

사전 조건

- 데스크톱 환경에는 최소 2.4GB의 메모리가 필요합니다. 따라서 적절한 성능을 보장하기 위해 유형 t2.medium 이상의 인스턴스를 사용하는 것이 좋습니다. 메모리가 부족한 인스턴스 유형의 예로는

t2.nano, t2.micro 및 t2.small이 있습니다. 이 제한은 메모리 요구 사항을 충족하지 않는 다른 인스턴스 유형뿐만 아니라 이 크기의 t3 및 t4 인스턴스에도 적용됩니다.

- 이 자습서에서는 릴리스 2023.7 이상을 실행하는 AL2023을 사용하여 인스턴스를 이미 시작했다고 가정합니다. 자세한 내용은 [Amazon EC2 AL2023](#) 및 [AL2023 업데이트](#) 페이지를 참조하세요.

설치

- GNOME 데스크톱 환경 및 관련 패키지를 설치합니다.

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

Note

그래픽 데스크톱 환경에 액세스하려면 Amazon DCV 또는 VNC와 같은 추가 소프트웨어를 설치하고 구성해야 합니다. 이러한 도구를 사용하면 네트워크를 통해 그래픽 사용자 인터페이스에 연결하고 상호 작용할 수 있습니다.

관련 주제

그래픽 데스크톱 환경에 대한 자세한 내용은 다음 설명서를 참조하세요.

- Amazon DCV 관리자 안내서의 [Amazon DCV란 무엇입니까?](#)
- [자습서: AL2023에서 TigerVNC 서버 구성](#)

자습서: AL2023에서 TigerVNC 서버 구성

다음 절차는 AL2023 인스턴스에서 VNC 서버를 설정하는 데 도움이 됩니다. VNC를 사용하면 보안 네트워크 연결을 통해 그래픽 데스크톱 환경에 원격으로 액세스하고 상호 작용할 수 있습니다.

내용

- [사전 조건](#)
- [1단계: 설치](#)
- [2단계: 구성](#)
- [3단계: VNC 클라이언트를 사용하여 연결](#)

- [\(선택 사항\) 부팅 시 서비스 시작](#)
- [\(선택 사항\) 유휴 잠금 화면 비활성화](#)
- [관련 주제](#)

사전 조건

- 이 자습서에서는 AL2023 인스턴스에 GNOME 데스크톱 환경을 이미 설치했다고 가정합니다. 자세한 내용은 [자습서: AL2023에 GNOME 데스크톱 환경 설치](#) 페이지를 참조하세요.
- 이 자습서에서는 SSH 포트 전달을 사용하여 VNC 서버에 액세스합니다. 키 쌍 설정에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [SSH를 사용하여 Linux 인스턴스에 연결](#) 섹션을 참조하세요.
- 다음 절차에서는 VNC 클라이언트 설치 프로세스를 안내하지 않습니다. 데스크톱 환경에 연결하고 상호 작용하려면 로컬 시스템에 VNC 클라이언트가 설치되어 있어야 합니다.

1단계: 설치

1. 인스턴스에 연결합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 단원을 참조하십시오.
2. AL2023용 TigerVNC 서버 패키지를 설치합니다.

-y 옵션을 사용하면 확인 여부를 묻지 않고 패키지를 설치합니다. 설치 전에 패키지를 확인하려면 이 옵션을 생략합니다.

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

2단계: 구성

1. 사용자가 VNC 암호를 구성했는지 확인합니다.

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

2. 사용자에게 표시 번호를 할당합니다.

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver.users
```

다음 구성을 추가합니다.

```
:1=ec2-user
```

Note

사용자에게 표시 번호를 할당할 수 있습니다. 이 예제에서는 디스플레이 :1을 사용합니다.

3. VNC 서버 구성 파일을 편집합니다.

```
[ec2-user ~]$ sudo vi /etc/tigervnc/vncserver-config-defaults
```

다음 구성을 추가합니다.

```
session=gnome
securitytypes=vncauth,tlsvnc
geometry=1920x1080
localhost
alwaysshared
```

Note

geometry 파라미터를 사용하여 디스플레이의 해상도를 변경할 수 있습니다. 이 예제에서는 1920x1080을 사용합니다.

4. VNC 서버를 시작합니다. 이 프로세스는 인스턴스를 다시 시작할 때마다 반복해야 합니다. 이 서비스를 시작하는 프로세스를 자동화하려면 아래 선택적 섹션을 참조하세요.

```
[ec2-user ~]$ sudo systemctl start vncserver@:1
```

Important

vncserver 서비스를 시작할 때 @ 다음 부분은 /etc/tigervnc/vncserver.users 파일의 사용자에게 대해 설정된 표시 번호와 일치해야 합니다.

이 단계를 수행한 후 로컬 시스템에서 SSH 터널을 생성하고 VNC 클라이언트를 사용하여 연결할 수 있습니다.

3단계: VNC 클라이언트를 사용하여 연결

VNC 서버는 클라이언트 연결을 위해 TCP 소켓을 노출합니다. 보안 그룹을 통해 VNC 포트를 직접 노출할 수 있지만, 이 자습서에서는 로컬 시스템과 EC2 인스턴스 간의 연결을 암호화하여 SSH 터널링을 보다 안전한 접근 방식으로 사용하는 방법을 보여줍니다. 터널을 통해 연결되면 이전 단계에서 구성한 암호를 사용하여 VNC 서버에 인증합니다. 보안 그룹에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 인스턴스에 대한 보안 그룹 변경](#) 섹션을 참조하세요.

1. 로컬 시스템에서 SSH 터널을 생성합니다.

```
$ ssh -i <keypair> -L 5901:localhost:5901 ec2-user@<address>
```

Note

<keypair>를 SSH 키의 경로로 바꾸고 <address>를 인스턴스의 퍼블릭 IP 또는 DNS 이름으로 바꿉니다. 포트는 vncserver를 시작하는 데 사용된 표시 번호에 따라 변경됩니다. 예를 들어 디스플레이 :1은 포트 5901을 사용하고 디스플레이 :2는 포트 5902를 사용합니다.

2. VNC 클라이언트를 사용하여 이전에 설정한 VNC 암호로 localhost:5901 또는 127.0.0.1:5901에 연결합니다.

Important

VNC를 사용하는 동안 SSH 터널을 열어 둡니다. SSH 터널이 열려 있지 않으면 VNC 클라이언트를 사용하여 데스크톱 환경을 보고 상호 작용할 수 없습니다.

(선택 사항) 부팅 시 서비스 시작

VNC를 정기적으로 사용하려는 경우 인스턴스가 부팅될 때 VNC 서버가 자동으로 시작되도록 구성할 수 있습니다. 따라서 인스턴스를 다시 시작할 때마다 VNC 서버를 수동으로 시작할 필요가 없습니다.

이 구성을 사용하면 인스턴스가 스타트업 프로세스를 완료하는 즉시 그래픽 데스크톱 환경을 준비하고 액세스할 수 있습니다.

- 부팅 시 시작하도록 서비스를 구성합니다.

```
[ec2-user ~]$ sudo systemctl enable vncserver@:1
```

Important

vncserver 서비스를 활성화할 때 @ 다음 부분은 /etc/tigervnc/vncserver.users 파일의 사용자에게 대해 설정된 표시 번호와 일치해야 합니다. 또한 enable 이후 --now 인수를 전달하여 서비스를 즉시 시작할 수 있습니다.

이 단계를 수행한 후에는 인스턴스를 재부팅할 때마다 vncserver를 더 이상 시작할 필요가 없습니다.

(선택 사항) 유휴 잠금 화면 비활성화

- 사용자가 장기간 비활성 상태일 때 잠금 화면을 비활성화하려면 유휴 지연을 0으로 설정합니다.

```
[ec2-user ~]$ gsettings set org.gnome.desktop.session idle-delay 0
```

관련 주제

그래픽 데스크톱 환경에 대한 자세한 내용은 다음 설명서를 참조하세요.

- [자습서: AL2023에 GNOME 데스크톱 환경 설치](#)
- Amazon DCV 관리자 안내서의 [Amazon DCV란 무엇입니까?](#)

AL2023 커널에서 Multi-Gen LRU(MGLRU) 사용

[Multi-Gen LRU](#)는 Linux 커널의 최신 페이지 재생 알고리즘으로, 메모리 부족 시 메모리 관리 성능을 개선하도록 설계되었습니다. 시스템에서 메모리 부족이 발생할 때 회수할 메모리 페이지를 결정하는 데 사용되는 기존의 가장 오래 전에 사용된(LRU) 메커니즘을 대체합니다.

기존 LRU 메커니즘은 2개 목록 모델(활성 및 비활성)을 사용하여 페이지 사용량을 추적하므로 대규모 작업 세트가 있는 최신 워크로드에서는 비효율적일 수 있습니다. MGLRU는 이를 여러 페이지의 '세대'로 대체하므로 커널이 더 세분화된 노후화 정보를 기반으로 더 스마트한 결정을 내릴 수 있습니다.

MGLRU의 이점은 다음과 같습니다.

- 더 나은 회수 결정: 콜드(미사용) 페이지를 더 정확하게 식별합니다.
- 지연 시간 단축 및 처리량 향상: 특히 주소 공간이 크거나 동시 프로세스가 많은 워크로드에 적합합니다.
- 캐시 보존 개선: 최근에 사용된 페이지는 조기에 제거될 가능성이 낮습니다.
- 확장 가능하고 잠금 효율적인 설계: 많은 CPU가 있는 시스템에서 더 나은 성능을 발휘합니다.

구성 및 튜닝

커널 구성 CONFIG_LRU_GEN은 AL2023 커널에서 활성화됩니다. 이렇게 하면 MGLRU에서 컴파일되지만 기본적으로 활성화되지는 않습니다.

/sys/kernel/mm/lru_gen/enabled 파일을 사용하여 MGLRU를 활성화하고 조정할 수 있습니다. 값은 비트 마스크입니다. 일부 구성 요소에 바람직하지 않은 부작용이 없는 한 모든 구성 요소를 활성화하는 것이 좋습니다.

Bit	구성 요소
0	다중 세대 LRU의 기본 스위치입니다.
1	MMU가 설정할 때 리프 페이지 테이블 항목의 액세스된 비트를 큰 배치로 지웁니다(예: x86). 이 동작은 이론적으로 잠금 경합(mmap_lock)을 악화시킬 수 있습니다. 비활성화되면 다중 세대 LRU는 핫 페이지를 연속으로 매핑하는 워크로드에서 약간의 성능 저하를 겪게 되며, 액세스한 비트는 더 적은 수의 더 큰 배치로 지울 수 있습니다.
2	MMU가 설정할 때(예: x86) 리프가 아닌 페이지 테이블 항목에서도 액세스된 비트를 지웁니다. 이 동작은 Intel 및 AMD 이외의 x86 품종에서는

Bit	구성 요소
	확인되지 않았습니다. 비활성화되면 다중 세대 LRU의 성능이 무시할 수 있는 수준으로 저하됩니다.
[yYnN]	위의 모든 구성 요소를 활성화/비활성화합니다.

MGLRU를 활성화하는 방법의 예제:

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

이렇게 하면 모든 구성 요소가 활성화됩니다.

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

자습서: AL2023에 LibreOffice 설치

LibreOffice는 단어 처리, 스프레드시트, 프레젠테이션, 다이어그램, 데이터베이스 및 공식 편집을 위한 애플리케이션을 포함하는 무료 오픈 소스 사무실 생산성 제품군입니다. Amazon Linux 2023(AL2023) 인스턴스에 LibreOffice를 설치하여 사용자에게 모든 기능을 갖춘 데스크톱 사무실 제품군을 제공할 수 있습니다.

LibreOffice는 문서 재단에서 AL2023에서 공식적으로 테스트하고 지원합니다. 자세한 내용은 [LibreOffice 64비트 진행 상황 및 Amazon Linux 2023 지원을](#) 참조하세요.

내용

- [사전 조건](#)
- [1단계: 시스템 아키텍처 결정](#)
- [2단계: LibreOffice RPM 패키지 다운로드](#)
- [3단계: 아카이브 추출](#)
- [4단계: RPM 패키지 설치](#)
- [5단계: 설치 확인](#)
- [LibreOffice 시작](#)

- [문제 해결](#)

사전 조건

- 원격 데스크톱 프로토콜(예: Amazon EC2 DCV)을 통해 구성되고 액세스할 수 있는 GNOME 그래픽 데스크톱 환경이 있는 AL2023을 실행하는 Amazon EC2 인스턴스입니다. 자세한 내용은 다음을 참조하세요.
 - [자습서: AL2023에 GNOME 데스크톱 환경 설치](#)
 - [Amazon DCV 서버 설정](#)
- sudo 또는 인스턴스의 루트 액세스.
- 설치를 위한 최소 1.5GB의 여유 디스크 공간.
- LibreOffice 패키지 또는 사전 다운로드되어 인스턴스로 전송된 패키지를 다운로드하기 위한 활성 인터넷 연결입니다.

1단계: 시스템 아키텍처 결정

다음 명령을 실행하여 프로세서 아키텍처를 식별합니다. 다운로드할 RPM 패키지를 결정합니다.

```
uname -p
```

출력은 다음 중 하나입니다.

출력	아키텍처	RPM 패키지 유형
x86_64	64비트 Intel/AMD	Linux_x86-64_rpm
aarch64	64비트 ARM(Graviton)	Linux_aarch64_rpm

2단계: LibreOffice RPM 패키지 다운로드

LibreOffice RPM 패키지를 다운로드하려면

1. [LibreOffice 다운로드 페이지](#)를 엽니다.
2. Linux(64비트)(rpm)를 운영 체제로 선택합니다.

3. 아키텍처에 적합한 버전(x86_64 또는 aarch64)을 선택합니다.
4. .tar.gz 아카이브를 인스턴스에 다운로드합니다. wget 또는 curl를 사용하여 터미널에서 직접 다운로드할 수 있습니다.

```
cd ~/Downloads
wget https://download.documentfoundation.org/libreoffice/stable/VERSION/
rpm/ARCHITECTURE/LibreOffice_VERSION_Linux_ARCHITECTURE.rpm.tar.gz
```

VERSION 및 *ARCHITECTURE*를 설치하려는 버전 번호 및 아키텍처로 바꿉니다(예: 25.2.5 및 x86_64).

Note

안정적인 최신 버전 번호를 찾으려면 [LibreOffice 다운로드 페이지](#)를 방문하세요.

3단계: 아카이브 추출

아카이브를 다운로드한 디렉터리로 이동하여 압축을 풉니다.

```
cd ~/Downloads
tar zxvf LibreOffice_VERSION_Linux_ARCHITECTURE.rpm.tar.gz
```

그러면와 비슷한 이름의 디렉터리가 생성됩니다
다LibreOffice_*VERSION*_Linux_*ARCHITECTURE*.rpm/.

4단계: RPM 패키지 설치

RPM 패키지를 설치하려면

1. 를 추출된 폴더 내의 RPMS/ 하위 디렉터리로 변경합니다.

```
cd LibreOffice_VERSION_Linux_ARCHITECTURE.rpm/RPMS/
```

2. 를 사용하여 모든 RPM 패키지를 설치합니다sudo rpm.

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

또는 종속성 확인dnf에를 사용할 수 있습니다.

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

5단계: 설치 확인

버전을 확인하여 LibreOffice가 성공적으로 설치되었는지 확인합니다.

```
libreoffice --version
```

예상 결과:

```
LibreOffice VERSION ...
```

LibreOffice 시작

터미널에서

다음 명령을 실행합니다.

```
libreoffice
```

libreoffice 명령을 사용할 수 없는 경우 바이너리 버전이 지정될 수 있습니다. 대신 버전이 지정된 이름을 사용하여 시작할 수 있습니다.

```
libreofficeMAJOR_VERSION
```

예를 들어 libreoffice25.2입니다.

특정 애플리케이션을 직접 시작하려면:

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

GNOME 데스크톱에서

설치 후 LibreOffice 애플리케이션이 GNOME 활동 메뉴에 나타납니다. 왼쪽 상단 모서리에서 활동을 선택하고 "LibreOffice"를 검색한 다음 원하는 애플리케이션을 선택합니다.

문제 해결

다음 섹션에서는 AL2023에서 LibreOffice를 설치하거나 실행할 때 발생하는 일반적인 문제를 해결하는 데 도움이 될 수 있습니다.

Java 런타임 환경 경고

일부 LibreOffice 기능(예: 기본 및 특정 확장)에는 Java 런타임 환경(JRE)이 필요합니다. Java 누락에 대한 경고가 표시되면 설치합니다.

```
sudo dnf install -y java-17-amazon-corretto-headless
```

그런 다음 LibreOffice에서 도구 > 옵션 > LibreOffice > 고급으로 이동하여 JRE가 감지되었는지 확인합니다.

LibreOffice 명령을 찾을 수 없음

설치 후 libreoffice 명령을 찾을 수 없는 경우 바이너리 버전이 지정될 수 있습니다. 실행해 보십시오.

```
libreofficeMAJOR_VERSION
```

예를 들어 libreoffice25.2입니다. 바이너리를 찾을 수도 있습니다.

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

추가 리소스

다른 문제가 발생하는 경우 Linux용 공식 LibreOffice 설치 설명서를 참조하세요.

- [LibreOffice Linux 설치 안내서](#)
- [LibreOffice 다운로드 페이지](#)

Amazon EC2 외부에서 Amazon Linux 2023 사용

Amazon Linux 2023 컨테이너 이미지는 호환되는 컨테이너 런타임 환경에서 실행됩니다. 컨테이너에서 Amazon Linux 2023을 사용하는 방법에 대한 자세한 내용은 [AL2023 컨테이너](#) 섹션을 참조하세요.

Amazon Linux 2023(AL2023)은 Amazon EC2에서 실행되지 않고 가상 게스트로 실행할 수도 있습니다. 현재 KVM(qcow2), VMware(OVA), Hyper-V(vhdx) 이미지를 사용할 수 있습니다.

Note

Amazon Linux 2023 이미지 구성은 Amazon Linux 2와 다릅니다.

[온프레미스에서 가상 머신으로 Amazon Linux 2 실행](#)하는 경우 AL2023 호환되도록 구성을 수정해야 합니다.

KVM, VMware 및 Hyper-V에서 사용할 Amazon Linux 2023 이미지 다운로드

KVM, VMware 및 Hyper-V에서 사용할 Amazon Linux 2023 디스크 이미지는 cdn.amazonlinux.com에서 다운로드할 수 있습니다.

Amazon EC2가 아닌 가상 환경에서 사용하기 위해 지원되는 Amazon Linux 2023 구성

이 섹션에서는 KVM, VMware 또는 Hyper-V처럼 Amazon EC2 가상화 환경이 아닌 환경에서 Amazon Linux 2023을 실행하기 위한 요구 사항을 다룹니다.

이 기본 [AL2023 시스템 요구 사항](#)은 Amazon EC2 가상 환경이 아닌 환경을 지원합니다. 다음 항목의 각 하이퍼바이저 환경은 지원되는 디바이스 모델 목록에 자세히 설명되어 있습니다.

KVM, VMware 및 Hyper-V는 다양한 구성 옵션을 제공하므로 보안, 성능 및 신뢰성 요구 사항에 맞게 주의해서 구성해야 합니다. 자세한 내용은 하이퍼바이저가 제공하는 설명서를 참조하세요.

주제

- [KVM에서 AL2023을 실행하기 위한 요구 사항](#)
- [VMware에서 AL2023을 실행하기 위한 요구 사항](#)
- [Hyper-V에서 Amazon Linux 2023 실행 요구 사항](#)

KVM에서 AL2023을 실행하기 위한 요구 사항

이 섹션에서는 KVM에서 AL2023을 실행하기 위한 요구 사항을 설명합니다. AL2023 KVM 이미지는 aarch64 및 x86-64 아키텍처에서 모두 사용할 수 있습니다. 이러한 요구 사항은 KVM 이미지의 기본 [AL2023 시스템 요구 사항](#) 외에 추가로 적용됩니다.

주제

- [KVM에서 AL2023을 실행하기 위한 KVM 호스트 요구 사항](#)
- [KVM에서 AL2023에 대한 디바이스 지원](#)
- [KVM에서 AL2023에 대한 부팅 모드\(UEFI 및 BIOS\)](#)
- [KVM에서 AL2023 실행 시 제한 사항](#)

KVM에서 AL2023을 실행하기 위한 KVM 호스트 요구 사항

KVM 이미지는 현재 Ubuntu 22.04.3 LTS를 실행하는 호스트에서 qemu 버전 6.2+dfsg-2ubuntu6.15를 사용하여 검증되었으며, x86-64용 q35 머신 유형과 aarch64용 virt 머신 유형을 사용합니다.

KVM에서 AL2023에 대한 디바이스 지원

AL2023 KVM 이미지(aarch64 및 x86-64 모두)로 테스트된 **qemu** 장치 모델은 다음과 같습니다.

- virtio-blk (virtio 블록 디바이스)
- virtio-scsi (디스크 장치가 있는 virtio SCSI 컨트롤러)
- virtio-net (virtio 네트워크 디바이스)
- ahci (가상 CD-ROM 드라이브에만 사용 가능)
- usb-storage (over xhci)

AL2023 KVM 이미지 검증에서 추가로 활성화되었지만 많이 테스트되지 않은 **qemu** 디바이스 모델은 다음과 같습니다.

- x86-64 VGA (qemu VGA)만 해당
- virtio-rng (가상 난수 생성기)
- 레거시 AT 키보드 및 PS/2 마우스 디바이스
- 레거시 시리얼 디바이스

KVM에서 AL2023에 대한 부팅 모드(UEFI 및 BIOS)

x86-64 이미지는 레거시 BIOS 및 UEFI 부팅 모드에서 테스트했습니다. aarch64 이미지는 UEFI 부팅 모드에서 테스트했습니다.

Note

기본적으로 UEFI 부팅 모드를 사용할 때 일부 가상 머신 관리자는 Microsoft 보안 부팅 키로 VM을 프로비저닝하여 보안 부팅을 활성화합니다. 이 구성으로는 AL2023 부팅은 불가능합니다.

AL2023 부트 로더는 Microsoft의 서명을 받지 않았으므로 VM은 UEFI 키 없이 프로비저닝하거나 보안 부팅용 AL2023 키를 사용하여 프로비저닝해야 합니다.

Important

KVM 이미지에 대한 보안 부팅 지원은 아직 검증되지 않았습니다.

KVM에서 AL2023 실행 시 제한 사항

KVM에서 AL2023을 실행할 때 알려진 몇 가지 제한 사항이 있습니다.

Note

나열되어 있는 지원되지 않는 기능 중 일부를 구현하는 코드가 AL2023 내에 존재할 수 있으며 올바르게 작동할 수 있습니다. 지원되지 않는 기능 목록은 현재 사용 가능한 기능과 향후 Amazon Linux 팀이 업데이트의 일부로 검증할 기능에 대해 정보에 입각한 결정을 내릴 수 있도록 제공됩니다.

KVM에서 AL2023 실행과 관련된 알려진 제한 사항

- KVM 게스트 에이전트는 현재 패키지로 제공되거나 지원되지 않습니다.
- CPU, 메모리 또는 기타 디바이스 유형의 핫 플러그 및 언플러그는 지원되지 않습니다.
- VM 최대 절전 모드는 지원되지 않습니다.
- VM 마이그레이션은 지원되지 않습니다.
- PCI 패스스루 또는 USB 패스스루와 같은 디바이스 패스스루는 지원되지 않습니다.

VMware에서 AL2023을 실행하기 위한 요구 사항

이 섹션에서는 VMware에서 AL2023을 실행하기 위한 요구 사항을 설명합니다. AL2023의 VMware 이미지는 x86-64 아키텍처에서만 사용할 수 있습니다. aarch64용 VMware 이미지는 사용할 수 없거나 지원되지 않습니다. 이러한 요구 사항은 VMware 이미지에 대한 기본 [AL2023 시스템 요구 사항](#) 외에 추가로 적용됩니다.

주제

- [VMware에서 AL2023을 실행하기 위한 VMware 호스트 요구 사항](#)
- [VMware에서 AL2023에 대한 디바이스 지원](#)
- [VMware에서 AL2023에 대한 부팅 모드\(UEFI 및 BIOS\) 지원](#)
- [VMware에서 AL2023 실행 시 제한 사항](#)

VMware에서 AL2023을 실행하기 위한 VMware 호스트 요구 사항

AL2023 VMware OVA 이미지는 현재 다음에서 검증되었습니다.

- Intel(R) Xeon(R) Platinum 8124M 프로세서를 사용하는 호스트에서 실행되는 VMware Workstation 17.5.0
- Intel(R) Xeon(R) Platinum 8275CL 프로세서를 사용하는 VMware vSphere 8.0

AL2023 VMware OVA 이미지는 머신 하드웨어 버전을 13으로 지정합니다.

VMware 머신 하드웨어 버전 13은 다음에서 지원을 받습니다.

- ESXi 6.5 이상
- VMware Workstation 14 이상

VMware에서 AL2023에 대한 디바이스 지원

AL2023 VMware OVA 이미지(**x86-64**만 해당)와 함께 사용하도록 테스트된 VMware 디바이스 모델은 다음과 같습니다.

- vmw_pvscsi(VMware 반가상화 SCSI 컨트롤러)
- vmxnet3(VMware 반가상화 네트워크 디바이스)
- ata_piix(가상 CD-ROM 드라이브에만 사용 가능한 레거시 IDE)

AL2023 VMware 이미지 검증에서 추가로 활성화되었지만 많이 테스트되지 않은 VMware 디바이스 모델은 다음과 같습니다.

- vmw_vmci 및 관련 vsock 인터페이스(VMware 게스트 에이전트를 위한 가상 소켓 전송)
- vmw_balloon 메모리 벌룬 디바이스
- VMware SVGA 컨트롤러
- 레거시 AT 키보드 및 PS/2 마우스 디바이스

VMware 게스트 에이전트 패키지(open-vm-tools)는 AL2023 VMware OVA 이미지에 기본적으로 설치되어 사용 가능합니다.

VMware에서 AL2023에 대한 부팅 모드(UEFI 및 BIOS) 지원

2023.3.20231211 릴리스부터 AL2023 VMware OVA 이미지는 레거시 BIOS 및 UEFI 부팅 모드 모두에서 검증되었습니다. OVA 기본 구성은 여전히 레거시 BIOS이지만 사용자가 변경할 수 있습니다.

Important

보안 부팅 지원에 UEFI가 필요하지만 VMware에서 실행되는 AL2023에 대해서는 아직 검증되지 않았습니다.

VMware에서 AL2023 실행 시 제한 사항

VMware에서 AL2023을 실행할 때 알려진 몇 가지 제한 사항이 있습니다.

Note

나열되어 있는 지원되지 않는 기능 중 일부를 구현하는 코드가 AL2023 내에 존재하며 올바르게 작동합니다. 지원되지 않는 기능 목록을 통해 고객은 현재 작업 시 사용 가능한 기능과 향후 Amazon Linux 팀이 업데이트할 기능을 확인할 수 있습니다.

VMware에서 AL2023 실행과 관련된 알려진 제한 사항

- UEFI 보안 부팅은 현재 VMware에서 AL2023으로 검증되지 않았습니다.
- CPU, 메모리 또는 기타 디바이스 유형의 핫 플러그 및 언플러그는 지원되지 않습니다.

- VM 최대 절전 모드는 지원되지 않습니다.
- VM 마이그레이션은 지원되지 않습니다.
- PCI 패스스루 또는 USB 패스스루와 같은 디바이스 패스스루는 지원되지 않습니다.

Hyper-V에서 Amazon Linux 2023 실행 요구 사항

이 섹션에서는 Hyper-V에서 Amazon Linux 2023을 실행하기 위한 요구 사항을 다룹니다. AL2023의 Hyper-V 이미지는 x86-64 아키텍처에서만 사용할 수 있습니다. aarch64용 Hyper-V 이미지는 현재 사용할 수 없거나 지원되지 않습니다.

이 섹션에서는 Hyper-V 이미지에 대한 기본 [AL2023 시스템 요구 사항](#) 외에 추가 요구 사항을 다룹니다.

주제

- [Hyper-V에서 Amazon Linux 2023을 실행하기 위한 Hyper-V 호스트 요구 사항](#)
- [Hyper-V에서 Amazon Linux 2023 디바이스 지원](#)
- [Hyper-V에서 Amazon Linux 2023 실행 시 제한 사항](#)

Hyper-V에서 Amazon Linux 2023을 실행하기 위한 Hyper-V 호스트 요구 사항

Hyper-V 기반 Amazon Linux 2023의 주요 검증은 EC2 c5.meta1 인스턴스에서 실행되는 Windows Server 2022에서 수행됩니다.

Hyper-V에서 Amazon Linux 2023 디바이스 지원

Amazon Linux 2023은 다음과 같은 가상화 하드웨어 세트를 사용하여 1세대 및 2세대 Hyper-V 가상 머신 모두에서 테스트됩니다.

- 1세대(레거시 BIOS 부팅) VM
- 2세대(UEFI 부팅 - 보안 부팅 없음) VM
- 다음 디바이스 모델은 AL2023 Hyper-V 이미지와 함께 사용하도록 테스트되었습니다.
 - 2세대 VM에서 루트 디스크 및 에뮬레이션된 CD-ROM 드라이브용 Hyper-V 가상 스토리지 hv_storvsc
 - 1세대 VM의 가상 CD-ROM 드라이브용 에뮬레이션 PIIX IDE ata_piix
 - Hyper-V 가상 이더넷 hv_netvsc
- 다음 디바이스 모델이 활성화되었지만 가볍게 테스트되었습니다.

- 1세대 VM의 레거시 VGA 텍스트 모드
- 2세대 VM의 UEFI 펌웨어 기반 프레임버퍼 `simplifiedrmfb`
- Hyper-V 풍선 `hv_balloon`
- Hyper-V HID/마우스 `hid_hyperv`
- 현재 AL2023에서는 다음 디바이스 모드가 활성화되지 않습니다.
 - Hyper-V PCI 패스스루
 - Hyper-V DRM 그래픽

Important

2세대 가상 머신의 경우 보안 부팅은 지원되지 않으므로 Amazon Linux 2023을 성공적으로 부팅하려면 가상 머신을 시작하기 전에 비활성화해야 합니다. Hyper-V는 현재 Microsoft 자체 키로 서명된 소프트웨어 구성 요소가 있는 보안 부팅만 지원하는 반면, Amazon Linux 부트로더는 Amazon 프라이빗 키로 서명됩니다. Hyper-V는 현재 타사 키 가져오기를 지원하지 않습니다.

Hyper-V에서 Amazon Linux 2023 실행 시 제한 사항

다음은 Hyper-V에서 Amazon Linux 2023을 실행할 때 발생하는 몇 가지 알려진 제한 사항입니다.

Note

나열되어 있는 지원되지 않는 기능 중 일부를 구현하는 코드가 AL2023 내에 존재하며 올바르게 작동합니다. 지원되지 않는 기능 목록을 통해 고객은 현재 작업 시 사용 가능한 기능과 향후 Amazon Linux 팀이 업데이트할 기능을 확인할 수 있습니다.

Hyper-V에서 AL2023 실행과 관련된 알려진 제한 사항

- UEFI 보안 부팅 모드는 현재 Hyper-V의 AL2023에서 지원되지 않거나 작동하지 않습니다.
- CPU, 메모리 또는 기타 디바이스 유형의 핫 플러그 및 언플러그는 지원되지 않습니다.
- 가상 머신(VM) 하이버네이션은 지원되지 않습니다.
- 가상 머신(VM) 마이그레이션은 지원되지 않습니다.
- PCI 패스스루 또는 USB 패스스루와 같은 디바이스 패스스루는 지원되지 않습니다.

Amazon EC2 외부에서 사용 시 Amazon Linux 2023 설정 및 **cloud-init** 구성

이 섹션에서는 KVM, VMware 또는 Hyper-V와 같이 Amazon EC2에서 직접 실행하지 않을 때 Amazon Linux 2023 가상 머신을 설정하고 구성하는 방법을 설명합니다.

기본적으로 Amazon Linux 2023 가상 머신 이미지에 사용자 암호 또는 ssh 키를 쓸 수 없으며 처음 검색된 네트워크 인터페이스에서 DHCP를 통해 네트워크 구성을 가져옵니다. 즉, 추가 구성 없이는 생성된 가상 머신에 연결할 방법이 없습니다.

따라서 가상 머신에 어떤 형태로든 구성을 제공해야 합니다. Amazon Linux에서 이를 수행하는 표준 메커니즘은 cloud-init 데이터 소스를 통해 이루어집니다.

Amazon Linux 2023은 다음 데이터 소스로 검증되었습니다.

NoCloud

cloud-init 구성 파일이 포함된 시드 ISO9660 이미지가 들어 있는 가상 CD-ROM을 통해 온프레미스 이미지를 구성하는 기존 방법입니다.

VMware

Amazon Linux 2023은 VMware guestinfo.userdata 및 guestinfo.metadata의 VMware 전용 데이터 소스를 통해 vSphere에서 실행되는 VMware 이미지를 구성할 수 있도록 추가 지원합니다.

Note

데이터 소스 구성은 Amazon Linux 2와 다를 수 있습니다. 구체적으로 설명하면, Amazon Linux 2023에서 systemd-networkd를 구성할 수 있으며 이를 위해 [cloud-init 네트워크 구성 설명서](#)에서 설명한 cloud-init “네트워킹 구성 버전 2”를 따라야 합니다.

Amazon Linux 2023에 포함된 cloud-init 버전의 cloud-init 구성 메커니즘 전체 설명서는 [업스트림 cloud-init 설명서](#)에서 있습니다.

KVM 및 VMware에서 Amazon Linux 2023 NoCloud (**seed.iso**) **cloud-init** 구성

이 섹션에서는 `seed.iso` 이미지를 생성하고 사용하여 KVM 또는 VMware에서 실행되는 Amazon Linux 2023을 구성하는 방법을 다룹니다. KVM 및 VMware 환경에는 [Amazon EC2 인스턴스 메타 데이터 서비스\(IMDS\)](#)가 없기 때문에 Amazon Linux 2023을 구성하는 대체 방법이 필요합니다. `seed.iso` 이미지를 제공하는 것이 그러한 방법 중 하나에 해당합니다.

`seed.iso` 부팅 이미지에는 네트워크 구성, 호스트 이름, 사용자 데이터 등 새 VM을 부팅하고 구성하는 데 필요한 기본 정보가 포함되어 있습니다.

Note

`seed.iso` 이미지에는 VM을 부팅하는 데 필요한 구성 정보만 있습니다. Amazon Linux 2023 운영 파일은 없습니다.

`seed.iso` 부팅 이미지를 생성하려면 최소 구성 파일 두 개가 필요하며 세 개가 필요한 경우도 있습니다.

meta-data

이 파일에는 일반적으로 가상 머신의 호스트 이름이 있습니다.

user-data

이 파일로 사용자 계정, 암호, ssh 키 페어 및/또는 액세스 메커니즘을 구성합니다. 기본적으로 Amazon Linux 2 KVM 및 VMware 이미지로 `ec2-user` 사용자 계정을 생성합니다. `user-data` 구성 파일을 사용하여 기본 사용자 계정의 암호 및/또는 키를 설정합니다.

network-config (선택 사항)

이 파일은 일반적으로 가상 시스템의 네트워크 구성을 제공하여 기본 구성을 재정의합니다. 기본 구성은 사용 가능한 첫 번째 네트워크 인터페이스의 DHCP를 사용하는 것입니다.

seed.iso 디스크 이미지 생성

1. Linux 또는 macOS 컴퓨터에서 `seedconfig`라는 이름의 새 폴더를 만들어 이 폴더로 이동합니다.

Note

Windows 또는 다른 운영 체제에서도 가능하지만 `seed.iso` 이미지를 생성하려면 `mkisofs`와 동일한 도구가 있어야 합니다.

2. meta-data 구성 파일을 생성합니다.

- a. meta-data라는 이름의 새로운 파일을 만듭니다.
- b. 원하는 편집기에서 meta-data 파일을 열어 다음 구문을 추가합니다. 여기서 `vm-hostname`을 VM 호스트 이름으로 바꾸어야 합니다.

```
#cloud-config
local-hostname: vm-hostname
```

- c. meta-data 구성 파일을 저장하고 닫습니다.

3. user-data 구성 파일을 생성합니다.

- a. user-data라는 이름의 새로운 파일을 만듭니다.
- b. 원하는 편집기에서 user-data 파일을 열고 필요에 따라 변경한 후 추가합니다.

```
#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. user-data 구성 파일에 사용자 계정을 더 추가할 수 있습니다.

추가 사용자 계정, 계정 액세스 메커니즘, 암호 및 키 페어를 지정할 수 있습니다. 지원하는 지 시어 대한 자세한 내용은 [업스트림 cloud-init 설명서](#)를 참조하세요.

- d. user-data 구성 파일을 저장하고 닫습니다.

4. (선택 사항) network-config 구성 파일 생성

- a. network-config라는 명칭의 새로운 파일을 만듭니다.

- b. 원하는 편집기에서 `network-config` 파일을 열고 다음 구문을 추가합니다. 여기서 다양한 IP 주소를 설정에 적합한 주소로 바꾸어야 합니다.

```
#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 네트워크 구성은 VM 구성에 따라 변경될 수 있는 인터페이스 이름을 지정하는 대신 인터페이스의 MAC 주소와 일치시키는 메커니즘을 제공합니다. 네트워크 구성을 위한 이 cloud-init 기능(및 기타 기능)은 [업스트림 cloud-init 네트워크 구성 버전2 설명서](#)에 자세히 설명되어 있습니다.

- c. `network-config` 구성 파일을 저장하고 닫습니다.
5. 이전 단계에서 만든 `meta-data`, `user-data` 및 `network-config`(옵션) 구성 파일로 `seed.iso` 디스크 이미지를 생성합니다.

`seed.iso` 디스크 이미지를 만들 운영 체제에 따라 다음 중 하나를 수행합니다.

- Linux 시스템에서 **mkisofs** 또는 **genisoimage** 같은 도구를 사용하여 `seed.iso` 파일을 생성합니다. `seedconfig` 폴더로 이동하여 다음 명령을 실행합니다.

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
```

- `network-config`를 사용하는 경우 **mkisofs** 간접 호출 시 포함시키세요.

```
$ mkisofs -output seed.iso -volid cidata -joliet -rock user-data meta-data
network-config
```

- macOS 시스템에서 **hdiutil** 같은 도구를 사용하여 완성된 `seed.iso` 파일을 만들 수 있습니다. **hdiutil**은 파일 목록이 아닌 경로 이름을 사용하므로 `network-config` 구성 파일의 생성 여부에 관계없이 동일한 간접 호출을 사용할 수 있습니다.

```
$ hdiutil makehybrid -o seed.iso -hfs -joliet -iso -default-volume-name cidata
seedconfig/
```

- 이제 생성된 `seed.iso` 파일을 가상 CD-ROM 드라이브를 통해 새 Amazon Linux 2023 가상 머신에 첨부하면 `cloud-init`가 처음 부팅 시 이를 찾아 시스템에 구성을 적용합니다.

VMware에서 AL2023에 대한 VMware guestinfo **cloud-init** 구성

VMware 환경에는 [Amazon EC2 인스턴스 메타데이터 서비스\(IMDS\)](#)가 없으므로 AL2023을 구성하는 대체 방법이 필요합니다. 이 섹션에서는 VMware vSphere에서 사용 가능한 `seed.iso` 가상 CD-ROM 드라이브에 대한 대체 구성 메커니즘을 사용하는 방법을 설명합니다.

이 구성 방법은 VMware extraconfig 메커니즘을 사용하여 `cloud-init`에 구성 데이터를 제공합니다. 다음 각 키에 해당 **keyname.encoding** 속성을 제공해야 합니다.

VMware extraconfig 메커니즘에 다음 키를 제공할 수 있습니다.

guestinfo.metadata

cloud-init 메타데이터가 포함된 JSON 또는 YAML

guestinfo.userdata

cloud-config 형식의 cloud-init 사용자 데이터가 포함된 YAML 설명서.

guestinfo.vendordata (선택 사항)

cloud-init 공급업체 데이터를 포함하는 YAML

해당 인코딩 속성(`guestinfo.metadata.encoding`, `guestinfo.userdata.encoding` 및 `guestinfo.vendordata.encoding`)은 다음을 포함합니다.

base64

속성의 콘텐츠는 base64로 인코딩되어 있습니다.

gzip+base64

base64로 인코딩된 후 속성 콘텐츠가 gzip으로 압축됩니다.

Note

seed.iso 방법은 별도(선택 사항)의 network-config 구성 파일을 지원하며, VMware guestinfo는 네트워킹 구성이 제공되는 방식이 다릅니다. 다음 섹션에서 추가 정보를 제공합니다.

명시적 네트워크 구성이 필요한 경우 다음 두 가지 YAML 또는 JSON 속성의 형태로 metadata에 포함되어야 합니다.

network

JSON 또는 YAML 형식으로 인코딩된 네트워크 구성을 포함합니다.

network.encoding

위 네트워크 구성 데이터의 인코딩을 포함합니다. cloud-init에서 지원되는 인코딩은 guestinfo 데이터에 지원되는 것과 동일한 base64 및 gzip+base64입니다.

Example VMware vSphere **govc** CLI 도구를 사용하여 **guestinfo**를 통해 구성 전달

1. [KVM 및 VMware에서 Amazon Linux 2023 NoCloud \(seed.iso\) cloud-init 구성](#)에 설명된 대로 meta-data, user-data 및 선택 사항인 network-config 구성 파일을 준비합니다.
2. 구성 파일을 VMware guestinfo에서 사용할 수 있는 형식으로 변환합니다.

```
# '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 표준 AMI에 설치된 패키지와 AL2023 KVM 이미지 비교

AL2023 표준 AMI에 있는 RPM과 AL2023 KVM 이미지에 있는 RPM 간의 비교입니다.

패키지	AMI	KVM
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-netw ork		1.2
amazon-rpm-config	228	228

패키지	AMI	KVM
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
boost-thread	1.75.0	1.75.0
bzip2	1.0.8	1.0.8

패키지	AMI	KVM
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
crypto-policies	20220428	20220428
crypto-policies-sc ripts	20220428	20220428
cryptsetup	2.6.1	2.6.1

패키지	AMI	KVM
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
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2
dracut	055	055

패키지	AMI	KVM
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
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188
elfutils-libelf	0.188	0.188

패키지	AMI	KVM
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
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

패키지	AMI	KVM
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
grub2-tools-minimal	2.06	2.06
grubby	8.40	8.40
gssproxy	0.8.4	0.8.4
gzip	1.12	1.12

패키지	AMI	KVM
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
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

패키지	AMI	KVM
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
less	608	608
libacl	2.3.1	2.3.1
libaio	0.3.111	0.3.111
libarchive	3.7.4	3.7.4

패키지	AMI	KVM
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
libeconf	0.4.0	0.4.0
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12

패키지	AMI	KVM
libfdisk	2.37.4	2.37.4
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.4.0
libkcapi-hmaccalc	1.4.0	1.4.0
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
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
libnhttp2	1.59.0	1.59.0

패키지	AMI	KVM
libnl3	3.5.0	3.5.0
libpath_utils	0.2.1	0.2.1
libpcap	1.10.1	1.10.1
libpipeline	1.5.3	1.5.3
libpkgconf	1.8.0	1.8.0
libpsl	0.21.1	0.21.1
libpwquality	1.4.4	1.4.4
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.14.5
libreport-filesystem	2.15.2	2.15.2
libseccomp	2.5.3	2.5.3
libselinux	3.4	3.4
libselinux-utils	3.4	3.4
libsemanage	3.4	3.4
libsepol	3.4	3.4
libsigsegv	2.13	2.13
libsmartcols	2.37.4	2.37.4
libsolv	0.7.22	0.7.22
libss	1.46.5	1.46.5
libsss_certmap	2.9.4	

패키지	AMI	KVM
libsss_idmap	2.9.4	2.9.4
libsss_nss_idmap	2.9.4	2.9.4
libsss_sudo	2.9.4	
libstdc++	11.4.1	11.4.1
libstoragemgmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21
libtirpc	1.3.3	1.3.3
libunistring	0.9.10	0.9.10
libuser	0.63	0.63
libutempter	1.2.1	1.2.1
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

패키지	AMI	KVM
libyam1	0.2.5	0.2.5
libzstd	1.5.5	1.5.5
linux-firmware-whe nce	20210208(noarch)	20210208(noarch)
lm_sensors-libs	3.6.0	3.6.0
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsof	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1(x86_64)	2.1(x86_64)
mpfr	4.1.0	4.1.0
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

패키지	AMI	KVM
newt	0.52.21	0.52.21
nfs-utils	2.5.4	2.5.4
npth	1.6	1.6
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-softokn	3.90.0	3.90.0
nss-softokn-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14
ocaml-srpm-macros	6	6
oniguruma	6.9.7.1	
openblas-srpm-macros	2	2
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1
openssl	3.0.8	3.0.8
openssl-libs	3.0.8	3.0.8

패키지	AMI	KVM
openssl-pkcs11	0.4.12	0.4.12
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-libs	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50
perl-Class-Struct	0.66	0.66
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

패키지	AMI	KVM
perl-File-Basename	2.85	2.85
perl-File-Path	2.18	2.18
perl-File-stat	1.09	1.09
perl-File-Temp	0.231.100	0.231.100
perl-Getopt-Long	2.52	2.52
perl-Getopt-Std	1.12	1.12
perl-HTTP-Tiny	0.078	0.078
perl-if	0.60.800	0.60.800
perl-interpreter	5.32.1	5.32.1
perl-IO	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14

패키지	AMI	KVM
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08
perl-Term-ANSIColor	5.01	5.01
perl-Term-Cap	1.17	1.17
perl-Text-ParseWords	3.30	3.30
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300
perl-vars	1.05	1.05
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0

패키지	AMI	KVM
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
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	

패키지	AMI	KVM
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-jjsonschema	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

패키지	AMI	KVM
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
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

패키지	AMI	KVM
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
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

패키지	AMI	KVM
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
sssd-common	2.9.4	
sssd-kcm	2.9.4	
sssd-nfs-idmap	2.9.4	

패키지	AMI	KVM
strace	6.8	6.8
sudo	1.9.15	1.9.15
sysctl-defaults	1.0	1.0
sysstat	12.5.6	12.5.6
systemd	252.23	252.23
systemd-libs	252.23	252.23
systemd-networkd	252.23	252.23
systemd-pam	252.23	252.23
systemd-resolved	252.23	252.23
systemd-udev	252.23	252.23
system-release	2023.6.20241031	2023.6.20241031
systemtap-runtime	4.8	4.8
tar	1.34	1.34
tbb	2020.3	2020.3
tcpdump	4.99.1	4.99.1
tcsh	6.24.07	6.24.07
time	1.9	1.9
traceroute	2.1.3	2.1.3
tzdata	2024a	2024a
unzip	6.0	6.0

패키지	AMI	KVM
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
xz-libs	5.2.5	5.2.5
yum	4.14.0	4.14.0
zip	3.0	3.0

패키지	AMI	KVM
zlib	1.2.11	1.2.11
zram-generator	1.1.2	
zram-generator-defaults	1.1.2	
zstd	1.5.5	1.5.5

Amazon Linux 2023 표준 AMI에 설치된 패키지와 AL2023 VMware OVA 이미지 비교

AL2023 VMware OVA 이미지에 있는 RPM과 AL2023 표준 AMI에 있는 RPM 간의 비교입니다.

패키지	AMI	VMware OVA
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.5.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.6.20241031
amazon-linux-repo-s3	2023.6.20241031	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-network		1.2

패키지	AMI	VMware OVA
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
boost-filesystem	1.75.0	1.75.0
boost-system	1.75.0	1.75.0
boost-thread	1.75.0	1.75.0

패키지	AMI	VMware OVA
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
crypto-policies	20220428	20220428
crypto-policies-sc ripts	20220428	20220428

패키지	AMI	VMware OVA
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
dnf-utils	4.3.0	4.3.0
dosfstools	4.2	4.2

패키지	AMI	VMware OVA
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
elfutils-debuginfod-client	0.188	0.188
elfutils-default-yama-scope	0.188	0.188

패키지	AMI	VMware OVA
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
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

패키지	AMI	VMware OVA
glib2	2.74.7	2.74.7
glibc	2.34	2.34
glibc-all-langpacks	2.34	2.34
glibc-common	2.34	2.34
glibc-gconv-extra	2.34	2.34
glibc-locale-source	2.34	2.34
gmp	6.2.1	6.2.1
gnupg2-minimal	2.3.7	2.3.7
gnutls	3.8.0	3.8.0
go-srpm-macros	3.2.0	3.2.0
gpgme	1.15.1	1.15.1
gpm-libs	1.20.7	1.20.7
grep	3.8	3.8
groff-base	1.22.4	1.22.4
grub2-common	2.06	2.06
grub2-efi-x64-ec2	2.06	2.06
grub2-pc		2.06
grub2-pc-modules	2.06	2.06
grub2-tools	2.06	2.06
grub2-tools-minimal	2.06	2.06

패키지	AMI	VMware OVA
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
jq	1.7.1	

패키지	AMI	VMware OVA
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
less	608	608

패키지	AMI	VMware OVA
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
libeconf	0.4.0	0.4.0

패키지	AMI	VMware OVA
libedit	3.1	3.1
libev	4.33	4.33
libevent	2.1.12	2.1.12
libfdisk	2.37.4	2.37.4
libffi	3.4.4	3.4.4
libfido2	1.10.0	1.10.0
libgcc	11.4.1	11.4.1
libgcrypt	1.10.2	1.10.2
libgomp	11.4.1	11.4.1
libgpg-error	1.42	1.42
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1
libkcapi	1.4.0	1.4.0
libkcapi-hmaccalc	1.4.0	1.4.0
libldb	2.6.2	
libmaxminddb	1.5.2	1.5.2
libmetalink	0.1.3	0.1.3
libmnl	1.0.4	1.0.4
libmodulemd	2.13.0	2.13.0

패키지	AMI	VMware OVA
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-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

패키지	AMI	VMware OVA
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
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

패키지	AMI	VMware OVA
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-wheel	20210208	20210208
lm_sensors-libs	3.6.0	3.6.0
lmdb-libs	0.9.29	0.9.29
logrotate	3.20.1	3.20.1
lsof	4.94.0	4.94.0
lua-libs	5.4.4	5.4.4
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3
man-pages	5.10	5.10
microcode_ctl	2.1	2.1
mpfr	4.1.0	4.1.0

패키지	AMI	VMware OVA
nano	5.8	5.8
ncurses	6.2	6.2
ncurses-base	6.2	6.2
ncurses-libs	6.2	6.2
nettle	3.8	3.8
net-tools	2.0	2.0
newt	0.52.21	0.52.21
nfs-utils	2.5.4	2.5.4
npth	1.6	1.6
nspr	4.35.0	4.35.0
nss	3.90.0	3.90.0
nss-softokn	3.90.0	3.90.0
nss-softokn-freebl	3.90.0	3.90.0
nss-sysinit	3.90.0	3.90.0
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14
ocaml-srpm-macros	6	6
oniguruma	6.9.7.1	
openblas-srpm-macros	2	2

패키지	AMI	VMware OVA
openldap	2.4.57	2.4.57
openssh	8.7p1	8.7p1
openssh-clients	8.7p1	8.7p1
openssh-server	8.7p1	8.7p1
openssl	3.0.8	3.0.8
openssl-lib	3.0.8	3.0.8
openssl-pkcs11	0.4.12	0.4.12
open-vm-tools		12.3.0
os-prober	1.77	1.77
p11-kit	0.24.1	0.24.1
p11-kit-trust	0.24.1	0.24.1
package-notes-srpm-macros	0.4	0.4
pam	1.5.1	1.5.1
parted	3.4	3.4
passwd	0.80	0.80
pciutils	3.7.0	3.7.0
pciutils-lib	3.7.0	3.7.0
pcre2	10.40	10.40
pcre2-syntax	10.40	10.40
perl-Carp	1.50	1.50

패키지	AMI	VMware OVA
perl-Class-Struct	0.66	0.66
perl-constant	1.33	1.33
perl-DynaLoader	1.47	1.47
perl-Encode	3.15	3.15
perl-Errno	1.30	1.30
perl-Exporter	5.74	5.74
perl-Fcntl	1.13	1.13
perl-File-Basename	2.85	2.85
perl-File-Path	2.18	2.18
perl-File-stat	1.09	1.09
perl-File-Temp	0.231.100	0.231.100
perl-Getopt-Long	2.52	2.52
perl-Getopt-Std	1.12	1.12
perl-HTTP-Tiny	0.078	0.078
perl-if	0.60.800	0.60.800
perl-interpreter	5.32.1	5.32.1
perl-I0	1.43	1.43
perl-IPC-Open3	1.21	1.21
perl-libs	5.32.1	5.32.1
perl-MIME-Base64	3.16	3.16

패키지	AMI	VMware OVA
perl-mro	1.23	1.23
perl-overload	1.31	1.31
perl-overloading	0.02	0.02
perl-parent	0.238	0.238
perl-PathTools	3.78	3.78
perl-Pod-Escapes	1.07	1.07
perl-podlators	4.14	4.14
perl-Pod-Perldoc	3.28.01	3.28.01
perl-Pod-Simple	3.42	3.42
perl-Pod-Usage	2.01	2.01
perl-POSIX	1.94	1.94
perl-Scalar-List-Utils	1.56	1.56
perl-SelectSaver	1.02	1.02
perl-Socket	2.032	2.032
perl-srpm-macros	1	1
perl-Storable	3.21	3.21
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08
perl-Term-ANSIColor	5.01	5.01
perl-Term-Cap	1.17	1.17

패키지	AMI	VMware OVA
perl-Text-ParseWords	3.30	3.30
perl-Text-Tabs+Wrap	2021.0726	2021.0726
perl-Time-Local	1.300	1.300
perl-vars	1.05	1.05
pkgconf	1.8.0	1.8.0
pkgconf-m4	1.8.0	1.8.0
pkgconf-pkg-config	1.8.0	1.8.0
polycoreutils	3.4	3.4
polycoreutils-python-utils	3.4	
popt	1.18	1.18
procps-ng	3.3.17	3.3.17
protobuf-c	1.4.1	1.4.1
psacct	6.6.4	6.6.4
psmisc	23.4	23.4
publicsuffix-list-dafsa	20240212	20240212
python3	3.9.16	3.9.16
python3-attrs	20.3.0	20.3.0
python3-audit	3.0.6	3.0.6
python3-awscrt	0.19.19	0.19.19

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

패키지	AMI	VMware OVA
python3-jsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-toolkit	3.0.24	3.0.24
python3-pycparser	2.20	2.20
python3-pyrsistent	0.17.3	0.17.3

패키지	AMI	VMware OVA
python3-pyserial	3.4	3.4
python3-pysocks	1.7.1	1.7.1
python3-pytz	2022.7.1	2022.7.1
python3-pyyaml	5.4.1	5.4.1
python3-requests	2.25.1	2.25.1
python3-rpm	4.16.1.3	4.16.1.3
python3-ruamel-yaml	0.16.6	0.16.6
python3-ruamel-yaml-clib	0.1.2	0.1.2
python3-setools	4.4.1	4.4.1
python3-setuptools	59.6.0	59.6.0
python3-setuptools-wheel	59.6.0	59.6.0
python3-six	1.15.0	1.15.0
python3-systemd	235	235
python3-urllib3	1.25.10	1.25.10
python3-wcwidth	0.2.5	0.2.5
python-chevron	0.13.1	
python-srpm-macros	3.9	3.9
quota	4.06	4.06
quota-nls	4.06	4.06

패키지	AMI	VMware OVA
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
selinux-policy-targeted	38.1.45	38.1.45
setup	2.13.7	2.13.7
shadow-utils	4.9	4.9

패키지	AMI	VMware OVA
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
systemtap-runtime	4.8	4.8
tar	1.34	1.34
tbb	2020.3	2020.3

패키지	AMI	VMware OVA
tcpdump	4.99.1	4.99.1
tcsh	6.24.07	6.24.07
time	1.9	1.9
traceroute	2.1.3	2.1.3
tzdata	2024a	2024a
unzip	6.0	6.0
update-motd	2.2	2.2
userspace-rcu	0.12.1	0.12.1
util-linux	2.37.4	2.37.4
util-linux-core	2.37.4	2.37.4
vim-common	9.0.2153	9.0.2153
vim-data	9.0.2153	9.0.2153
vim-enhanced	9.0.2153	9.0.2153
vim-filesystem	9.0.2153	9.0.2153
vim-minimal	9.0.2153	9.0.2153
wget	1.21.3	1.21.3
which	2.21	2.21
words	3.0	3.0
xfsdump	3.1.11	3.1.11
xfstools	5.18.0	5.18.0

패키지	AMI	VMware OVA
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 표준 AMI에 설치된 패키지와 AL2023 Hyper-V 이미지 비교

AL2023 표준 AMI에 있는 RPM과 AL2023 Hyper-V 이미지에 있는 RPM 간의 비교입니다.

패키지	AMI	Hyper-V VHDX
acl	2.3.1	2.3.1
acpid	2.0.32	
alternatives	1.15	1.15

패키지	AMI	Hyper-V VHDX
amazon-chrony-config	4.3	
amazon-ec2-net-utils	2.4.1	
amazon-linux-onprem		1.2
amazon-linux-repo-cdn		2023.4.20240319
amazon-linux-repo-s3	2023.4.20240319	
amazon-linux-sb-keys	2023.1	2023.1
amazon-onprem-network		1.2
amazon-rpm-config	228	228
amazon-ssm-agent	3.2.2303.0	3.2.2303.0
at	3.1.23	3.1.23
attr	2.5.1	2.5.1
audit	3.0.6	3.0.6
audit-libs	3.0.6	3.0.6
aws-cfn-bootstrap	2.0	
awscli-2	2.14.5	2.14.5
basesystem	11	11
bash	5.2.15	5.2.15
bash-completion	2.11	2.11
bc	1.07.1	1.07.1

패키지	AMI	Hyper-V VHDX
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
cloud-utils-growpart	0.31	0.31
coreutils	8.32	8.32
coreutils-common	8.32	8.32

패키지	AMI	Hyper-V VHDX
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
diffutils	3.8	3.8
dnf	4.14.0	4.14.0
dnf-data	4.14.0	4.14.0

패키지	AMI	Hyper-V VHDX
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	
ec2-utils	2.2.0	
ed	1.14.2	1.14.2
efi-filesystem	5	5

패키지	AMI	Hyper-V VHDX
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
gawk	5.1.0	5.1.0
gdbm-libs	1.19	1.19
gdisk	1.0.8	1.0.8

패키지	AMI	Hyper-V VHDX
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
grub2-pc		2.06

패키지	AMI	Hyper-V VHDX
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
hyperv-tools		0
hypervvssd		0
info	6.7	6.7

패키지	AMI	Hyper-V VHDX
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
kernel-modules-ext ra-common		6.1.79
kernel-srpm-macros	1.0	1.0
kernel-tools	6.1.79	6.1.79

패키지	AMI	Hyper-V VHDX
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
libcbor	0.7.0	0.7.0
libcollection	0.7.0	0.7.0
libcom_err	1.46.5	1.46.5

패키지	AMI	Hyper-V VHDX
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
libibverbs	48.0	48.0
libidn2	2.3.2	2.3.2
libini_config	1.3.1	1.3.1

패키지	AMI	Hyper-V VHDX
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
libref_array	0.1.5	0.1.5
librepo	1.14.5	1.14.5
libreport-filessystem	2.15.2	2.15.2

패키지	AMI	Hyper-V VHDX
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
libstoragemgmt	1.9.4	1.9.4
libtalloc	2.3.4	
libtasn1	4.19.0	4.19.0
libtdb	1.4.7	
libtevent	0.13.0	
libtextstyle	0.21	0.21

패키지	AMI	Hyper-V VHDX
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
lua-srpm-macros	1	1
lz4-libs	1.9.4	1.9.4
man-db	2.9.3	2.9.3

패키지	AMI	Hyper-V VHDX
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
nss-util	3.90.0	3.90.0
ntsysv	1.15	1.15
numactl-libs	2.0.14	2.0.14

패키지	AMI	Hyper-V VHDX
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-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

패키지	AMI	Hyper-V VHDX
pcr2-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-interpretter	5.32.1	5.32.1
perl-IO	1.43	1.43
perl-IPC-Open3	1.21	1.21

패키지	AMI	Hyper-V VHDX
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
perl-subs	1.03	1.03
perl-Symbol	1.08	1.08

패키지	AMI	Hyper-V VHDX
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
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

패키지	AMI	Hyper-V VHDX
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
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

패키지	AMI	Hyper-V VHDX
python3-jsonpatch	1.21	1.21
python3-jsonpointer	2.0	2.0
python3-jjsonschema	3.2.0	3.2.0
python3-libcomps	0.1.20	0.1.20
python3-libdnf	0.69.0	0.69.0
python3-libs	3.9.16	3.9.16
python3-libselenium	3.4	3.4
python3-libsemanage	3.4	3.4
python3-libstorage mgmt	1.9.4	1.9.4
python3-lockfile	0.12.2	
python3-markupsafe	1.1.1	1.1.1
python3-netifaces	0.10.6	0.10.6
python3-oauthlib	3.0.2	3.0.2
python3-pip-wheel	21.3.1	21.3.1
python3-ply	3.11	3.11
python3-policycore utils	3.4	3.4
python3-prettytable	0.7.2	0.7.2
python3-prompt-too lkit	3.0.24	3.0.24

패키지	AMI	Hyper-V VHDX
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
python-chevron	0.13.1	
python-srpm-macros	3.9	3.9

패키지	AMI	Hyper-V VHDX
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	37.22	37.22
selinux-policy-targeted	37.22	37.22

패키지	AMI	Hyper-V VHDX
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
systemd-udev	252.16	252.16
system-release	2023.4.20240319	2023.4.20240319
systemtap-runtime	4.8	4.8

패키지	AMI	Hyper-V VHDX
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
words	3.0	3.0

패키지	AMI	Hyper-V VHDX
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 인스턴스 및 버전 식별

OS 이미지 또는 인스턴스의 Linux 배포와 배포 버전을 확인할 수 있어야 할 수 있습니다. Amazon Linux는 다른 Linux 배포판과 별도로 Amazon Linux를 식별하고 이미지의 Amazon Linux 릴리스를 식별하는 메커니즘을 제공합니다.

이 섹션에서는 사용할 수 있는 다양한 방법, 제한 사항을 다루고 몇 가지 사용 예제를 살펴봅니다.

주제

- [os-release 표준 사용](#)
- [Amazon Linux 특징](#)
- [OS 감지를 위한 코드 예제](#)

os-release 표준 사용

Amazon Linux는 Linux 배포판 식별을 위해 [os-release 표준](#)을 준수합니다. 이 파일은 운영 체제 식별 및 버전 정보에 대해 기계가 판독할 수 있는 정보를 제공합니다.

Note

표준은 /etc/os-release가 먼저 구문 분석되고 그 뒤에 /usr/lib/os-release가 오도록 지시합니다. 파일 이름 및 경로에 대한 표준을 따르도록 주의해야 합니다.

주제

- [주요 식별 차이점](#)
- [필드 유형: 기계가 판독할 수 있음 및 사람이 읽을 수 있음](#)
- [/etc/os-release 예제](#)
- [다른 배포판과 비교](#)

주요 식별 차이점

os-release는 /etc/os-release에서 찾을 수 있으며, 없는 경우 /usr/lib/os-release에서 찾을 수 있습니다. 자세한 내용은 [os-release 표준](#) 섹션을 참조하세요.

인스턴스가 Amazon Linux를 실행하는지 확인하는 가장 신뢰할 수 있는 방법은 `os-release`의 ID 필드를 확인하는 것입니다.

버전을 구분하는 가장 신뢰할 수 있는 방법은 `os-release`에서 `VERSION_ID` 필드를 확인하는 것입니다.

- Amazon Linux AMI: `VERSION_ID`는 날짜 기반 버전 포함(예: 2018.03)
- AL2: `VERSION_ID`="2"
- AL2023: `VERSION_ID`="2023"

Note

`VERSION_ID`는 프로그래밍 방식으로 사용하기 위한 기계가 판독할 수 있는 필드인 반면, `PRETTY_NAME`은 사용자에게 표시하도록 설계되었습니다. 필드 유형에 대한 자세한 내용은 [the section called “필드 유형”](#) 섹션을 참조하세요.

필드 유형: 기계가 판독할 수 있음 및 사람이 읽을 수 있음

`/etc/os-release` 파일(또는 `/etc/os-release`가 존재하지 않는 경우 `/usr/lib/os-release`)에는 프로그래밍 방식으로 사용하기 위한 기계가 판독할 수 있는 필드와 사용자에게 표시하기 위해 사람이 읽을 수 있는 필드라는 두 가지 유형의 필드가 포함되어 있습니다.

기계가 판독할 수 있는 필드

이러한 필드는 표준화된 형식을 사용하며 스크립트, 패키지 관리자 및 기타 자동화된 도구로 처리하기 위한 것입니다. 소문자, 숫자 및 제한된 구두점(마침표, 밑줄 및 하이픈)만 포함합니다.

- ID - 운영 체제 식별자입니다. Amazon Linux는 모든 버전에서 `amzn`을 사용하여 Debian(`debian`), Ubuntu(`ubuntu`) 또는 Fedora(`fedora`)와 같은 다른 배포판과 구별합니다.
- `VERSION_ID` - 프로그래밍 방식 사용을 위한 운영 체제 버전(예: 2023)
- `ID_LIKE` - 관련 배포판의 공백으로 구분된 목록(예: `fedora`)
- `VERSION_CODENAME` - 스크립트의 릴리스 코드 이름(예: `karoo`)
- `VARIANT_ID` - 프로그래밍 방식의 결정을 위한 변형 식별자
- `BUILD_ID` - 시스템 이미지의 빌드 식별자
- `IMAGE_ID` - 컨테이너화된 환경의 이미지 식별자

- PLATFORM_ID - 플랫폼 식별자(예: platform:al2023)

사람이 읽을 수 있는 필드

이러한 필드는 사용자에게 표시하기 위한 것이며 공백, 대소문자 혼합 및 설명 텍스트를 포함할 수 있습니다. 사용자 인터페이스에 운영 체제 정보를 제공할 때 사용해야 합니다.

- NAME - 표시할 운영 체제 이름(예: Amazon Linux)
- PRETTY_NAME - 표시할 버전이 있는 전체 운영 체제 이름(예: Amazon Linux 2023.8.20250721)
- VERSION - 사용자 프레젠테이션에 적합한 버전 정보
- VARIANT - 표시할 변형 또는 에디션 이름(예: Server Edition)

기타 정보 필드

이러한 필드는 운영 체제에 대한 추가 메타데이터를 제공합니다.

- HOME_URL - 프로젝트 홈페이지 URL
- DOCUMENTATION_URL - 설명서 URL
- SUPPORT_URL - 지원 정보 URL
- BUG_REPORT_URL - 버그 보고 URL
- VENDOR_NAME - 공급업체 이름
- VENDOR_URL - 공급업체 URL
- SUPPORT_END - YYYY-MM-DD 형식의 지원 종료일
- CPE_NAME - 공통 플랫폼 열거 식별자
- ANSI_COLOR - 터미널 표시용 ANSI 색상 코드

Amazon Linux를 프로그래밍 방식으로 식별해야 하는 스크립트 또는 애플리케이션을 작성할 때는 ID 및 VERSION_ID 같이 기계가 판독할 수 있는 필드를 사용합니다. 운영 체제 정보를 사용자에게 표시할 때는 PRETTY_NAME 같이 사람이 읽을 수 있는 필드를 사용합니다.

/etc/os-release 예제

/etc/os-release 파일 콘텐츠는 Amazon Linux 버전마다 다릅니다.

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/"

```

다른 배포판과 비교

Amazon Linux가 더 광범위한 Linux 에코시스템에 어떻게 적합한지 이해하려면 `/etc/os-release` 형식을 다른 주요 배포판과 비교하세요.

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
```

기계가 판독할 수 있는 필드가 배포판 전반에서 일관된 식별을 제공하는 방법을 확인합니다.

- ID - 운영 체제 고유 식별: Amazon Linux의 경우 `amzn`, Fedora의 경우 `fedora`, Debian의 경우 `debian`, Ubuntu의 경우 `ubuntu`
- ID_LIKE - 배포 관계 표시: Amazon Linux는 `fedora(AL2023)` 또는 `centos rhel fedora(AL2)`를 사용하고 Ubuntu는 Debian의 전통을 나타내는 `debian`를 표시

- `VERSION_ID` - 기계 구문 분석 가능한 버전 정보 제공: AL2023의 경우 2023, Fedora의 경우 42, Debian의 경우 12, Ubuntu의 경우 24.04

반면 사람이 읽을 수 있는 필드는 사용자에게 표시되도록 설계되었습니다.

- `NAME` - 사용자 친화적인 OS 이름: Amazon Linux, Fedora Linux, Debian GNU/Linux, Ubuntu
- `PRETTY_NAME` - 버전이 다음에 해당하는 전체 표시 이름: Amazon Linux 2023.8.20250721, Fedora Linux 42 (Container Image), Debian GNU/Linux 12 (bookworm), Ubuntu 24.04.2 LTS
- `VERSION` - 코드 이름 또는 릴리스 유형과 같은 추가 컨텍스트가 있는 사람이 읽을 수 있는 버전

교차 플랫폼 스크립트를 작성할 때는 항상 로직 및 결정에 기계가 판독할 수 있는 필드(`ID`, `VERSION_ID`, `ID_LIKE`)를 사용하고 사용자에게 정보를 표시할 때만 사람이 읽을 수 있는 필드(`PRETTY_NAME`, `NAME`)를 사용합니다.

Amazon Linux 특징

Amazon Linux와 해당 버전을 식별하는 데 사용할 수 있는 Amazon Linux와 관련된 일부 파일이 있습니다. 새 코드는 배포판 간 호환을 위해 [/etc/os-release](#) 표준을 사용해야 합니다. Amazon Linux 특정 파일은 사용하지 않는 것이 좋습니다.

주제

- [/etc/system-release 파일](#)
- [이미지 식별 파일](#)
- [Amazon Linux 특정 파일의 예제](#)

/etc/system-release 파일

Amazon Linux에는 설치된 현재 릴리스를 지정하는 `/etc/system-release` 파일이 포함되어 있습니다. 이 파일은 패키지 관리자를 사용하여 업데이트되며 Amazon Linux는 `system-release` 패키지의 일부입니다. Fedora와 같은 일부 다른 배포판에도 이 파일이 있지만 Ubuntu와 같은 Debian 기반 배포판에는 없습니다.

Note

/etc/system-release 파일에는 사람이 읽을 수 있는 문자열이 포함되어 있으므로 OS 또는 릴리스를 식별하는 데 프로그래밍 방식으로 사용해서는 안 됩니다. 대신 /etc/os-release의 기계가 판독할 수 있는 필드(또는 /etc/os-release가 존재하지 않는 경우 /usr/lib/os-release)를 사용합니다.

Amazon Linux에는 /etc/system-release의 기계가 판독할 수 있는 버전도 포함되어 있으며 /etc/system-release-cpe 파일의 CPE(Common Platform Enumeration) 사양을 따릅니다.

이미지 식별 파일

각 Amazon Linux 이미지에는 Amazon Linux 팀에서 생성한 원본 이미지에 대한 추가 정보를 제공하는 고유한 /etc/image-id 파일이 포함되어 있습니다. 이 파일은 Amazon Linux에만 해당되며 Debian, Ubuntu 또는 Fedora와 같은 다른 Linux 배포판에서는 찾을 수 없습니다. 이 파일에는 이미지에 대한 다음 정보가 포함되어 있습니다.

- image_name, image_version, image_arch - 이미지 생성에 사용한 빌드 레시피의 값입니다.
- image_stamp - 이미지 생성 중에 생성된 고유한 임의의 16진수 값입니다.
- image_date - YYYYMMDDhhmmss 형식으로 표시되는 이미지 생성 시간(UTC)입니다.
- recipe_name, recipe_id - 이미지 생성에 사용한 빌드 레시피의 이름과 ID입니다.

Amazon Linux 특정 파일의 예제

다음 섹션에서는 Amazon Linux의 각 메이저 버전에 대한 Amazon Linux별 식별 파일의 예제를 제공합니다.

Note

실제 코드에서 /etc/os-release 파일이 없는 경우 /usr/lib/os-release를 사용해야 합니다.

AL2023

다음 예제에서는 AL2023의 식별 파일을 보여줍니다.

AL2023용 /etc/image-id의 예제:

```
[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"
```

AL2023용 /etc/system-release의 예제:

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

```
Amazon Linux release 2023.8.20250721 (Amazon Linux)
```

AL2

다음 예제에서는 AL2의 식별 파일을 보여줍니다.

AL2용 /etc/image-id의 예제:

```
[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"
```

AL2용 /etc/system-release의 예제:

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

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

Amazon Linux AMI

다음 예제에서는 Amazon Linux AMI의 식별 파일을 보여줍니다.

Amazon Linux AMI용 `/etc/image-id`의 예제:

```
[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 AMI용 `/etc/system-release`의 예제:

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

```
Amazon Linux AMI release 2018.03
```

OS 감지를 위한 코드 예제

다음 예제에서는 `/etc/os-release`(또는 `/etc/os-release`가 존재하지 않는 경우 `/usr/lib/os-release`) 파일을 사용하여 운영 체제 및 버전을 프로그래밍 방식으로 감지하는 방법을 보여줍니다. 이 예제에서는 Amazon Linux와 다른 배포를 구분하는 방법과 `ID_LIKE` 필드를 사용하여 배포판 제품군을 결정하는 방법을 보여줍니다.

아래 스크립트는 여러 프로그래밍 언어로 구현되며 각 구현은 동일한 출력을 생성합니다.

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"
    esac
fi

```

```

        ;;
    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
            except IOError:
                continue

    return os_release_data

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";
    }
}

```

다른 시스템에서 실행하면 스크립트는 다음 출력을 생성합니다.

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)
```


파일 시스템 레이아웃

이 섹션에서는 인스턴스 또는 AL2023 기반 컨테이너에 고유할 수 있는 세부 정보를 포함하여 AL2023 시스템의 파일 시스템 레이아웃을 다룹니다. 자세한 내용은 `file-hierarchy(7)` man 페이지를 참조하세요.

주제

- [/\(루트 디렉터리\)](#)
- [/boot\(커널, initramfs 등\)](#)
- [/etc\(시스템 구성\)](#)
- [/home\(사용자 홈 디렉터리\)](#)
- [/root\(root 사용자 홈 디렉터리\)](#)
- [/srv\(서버 페이로드\)](#)
- [/tmp\(작은 임시 파일\)](#)
- [/run\(런타임 데이터\)](#)
- [/usr\(시스템 리소스\)](#)
- [/var\(영구 변수 시스템 데이터\)](#)

/(루트 디렉터리)

기본적으로 AL2023 이미지는 쓰기 가능한 /로 구성되므로 권한이 있는 사용자가 새 파일과 디렉터리를 생성할 수 있습니다.

다른 경로 또는 이미지를 사용하여 해당 `systemd` 서비스에 대해 /로 표시되도록 서비스를 구성하고 모든 경로에 액세스 제한을 적용할 수 있습니다.

Note

`systemd` 서비스에 대한 액세스 권한을 제한하도록 서비스를 구성하는 것이 가장 좋습니다. 여기에는 해당 서비스에 대해 /를 읽기 전용으로 만드는 `ReadOnlyPaths=/` 지시문 사용이 포함될 수 있습니다.

`systemd`를 사용하여 서비스에 대한 시스템 액세스를 제한하는 방법에 대한 자세한 내용은 `systemd.exec(5)` man 페이지를 참조하세요.

/boot(커널, initramfs 등)

기본적으로 부팅 가능한 AL2023 이미지는 root 파일 시스템에 있는 /boot로 구성됩니다. /boot 경로는 부팅 가능한 이미지에만 관련되므로 AL2023 컨테이너 이미지에서는 사용되지 않습니다.

이 디렉터리에는 Linux 커널 및 initramfs와 같이 AL2023이 부팅하는 데 필요한 파일이 있습니다. 이 디렉터리의 내용은 OS와 함께 제공된 도구를 사용해서만 조작해야 합니다.

/boot/efi(EFI 시스템 파티션)

기본적으로 부팅 가능한 AL2023 이미지는 /boot/efi에 탑재되는 EFI System 파티션으로 구성됩니다. 이 파일 시스템은 OS에서 관리하며 시스템 부팅에 중요한 코드와 구성을 포함합니다.

이 경로는 컨테이너 이미지와 관련이 없습니다.

/etc(시스템 구성)

AL2023의 /etc 디렉터리에는 시스템별 구성이 포함되어 있습니다. 기본적으로 AL2023 이미지는 root 파일 시스템에서 /etc와 함께 제공되며 권한이 있는 사용자가 쓸 수 있습니다.

Note

애플리케이션(systemd 포함)은 [/etc\(시스템 구성\)](#)에 구성을 배치하여 재정의할 수 있도록 [/usr\(시스템 리소스\)](#)에 따라 기본 구성을 유지하는 것이 일반적입니다.

이러한 애플리케이션의 경우 [/usr\(시스템 리소스\)](#)에서 기본 구성을 재정의하지 않고 /etc에서 파일을 변경하면 패키지가 업데이트될 때 변경 사항이 재정의될 수 있습니다.

/home(사용자 홈 디렉터리)

일반 사용자는 /home 아래에 홈 디렉터리가 있지만, 소프트웨어는 /home/\$USER 같은 패턴에 의존하지 않고 항상 사용자당 \$HOME 환경 변수를 찾아야 합니다.

기본적으로 AL2023 이미지는 root 파일 시스템에 /home이 있지만 소프트웨어는 이를 사용해서는 안 됩니다. OS를 별도의 파일 시스템이 되도록 /home으로 구성하여 나중에 부팅하는 동안 또는 사용자가 시스템에 인증한 후에만 탑재하는 것이 완벽하게 유효합니다.

root 사용자 홈 디렉터리는 /home에 있지 않고 /home 파일 시스템을 탑재할 수 없는 경우 사용할 수 있도록 [/root\(root 사용자 홈 디렉터리\)](#)에 있습니다.

Note

/home에 대한 쓰기 액세스 권한이 필요하지 않은 systemd 서비스를 ProtectHome=read-only 지시문으로 구성하는 것이 모범 사례입니다. 이 옵션을 사용하면 /home, /root 및 /run/user가 서비스에 대해 읽기 전용으로 설정됩니다.

/home에 액세스할 필요가 없는 서비스를 ProtectHome=tmpfs 지시문으로 구성하는 것 역시 모범 사례입니다. 이 지시문에 따라 /home, /root 및 /run/user가 빈 읽기 전용 tmpfs 파일 시스템인 샌드박스에서 서비스를 실행하게 됩니다.

systemd를 사용하여 서비스에 대한 시스템 액세스를 제한하는 방법에 대한 자세한 내용은 `systemd.exec(5)` man 페이지를 참조하세요.

/root(root 사용자 홈 디렉터리)

root 사용자의 홈 디렉터리는 /root 디렉터리이며, 사용할 수 없는 파일 시스템에 [/home\(사용자 홈 디렉터리\)](#)이 있는 이벤트에 존재하도록 의도적으로 [/home\(사용자 홈 디렉터리\)](#)과 분리되어 있습니다.

systemd 서비스 구성 모범 사례는 /root에 대한 모범 사례와 [/home\(사용자 홈 디렉터리\)](#)에 대한 모범 사례와 동일합니다.

/srv(서버 페이로드)

/srv 디렉터리는 시스템 관리자가 관리하며 Amazon Linux 2023은 이 디렉터리의 구성 방식을 제한하지 않습니다.

/srv 디렉터리가 별도의 파일 시스템에 있도록 구성할 수 있으므로 나중에 부팅할 때만 사용할 수 있습니다.

/tmp(작은 임시 파일)

Note

Amazon Linux 2023은 기본적으로 root 파일 시스템의 경로가 아니라 /tmp이 tmpfs이므로 Amazon Linux 2와 다릅니다.

Note

컨테이너에서 실행되는 경우 일반적으로 /tmp가 tmpfs인지 또는 디스크의 경로인지, 실행 중인 정리 프로세스가 있는지 여부를 지정하는 컨테이너 런타임 구성에 해당합니다.

/tmp 디렉터리는 크기가 한정된 작은 임시 파일을 위한 것입니다. 기본적으로 AL2023은 크기 제한이 RAM의 50%이고 최대 100만 inodes인 tmpfs 파일 시스템으로 구성합니다.

애플리케이션은 /tmp보다 \$TMPDIR 환경 변수의 경로를 선호해야 합니다. 그런 다음 사용자는 \$TMPDIR 환경 변수를 설정하여 애플리케이션이 /tmp에 사용해야 하는 경로를 재정의할 수 있습니다.

더 큰 임시 파일의 경우 [/var/tmp](#)를 대신 사용해야 합니다.

Warning

/tmp는 공유되므로 임시 파일을 생성하는 안전한 방법을 사용하는 것이 중요합니다. 자세한 내용은 [/tmp 및 /var/tmp의 안전한 사용](#)에 대한 업스트림 systemd 설명서를 참조하세요.

Note

/tmp 및 [/var/tmp](#)가 호스트 또는 다른 서비스와 공유되지 않는 샌드박스에서 서비스를 실행하는 yes 또는 disconnected로 설정된 PrivateTmp= 지시문으로 systemd 서비스를 구성하는 것이 가장 좋습니다.

동일한 프라이빗 임시 디렉터리를 공유하도록 두 서비스를 구성하는 방법을 비롯한 자세한 내용은 `systemd.exec(5)` man 페이지를 참조하세요.

/tmp의 콘텐츠는 일반적으로 부팅 시 정리되며 사용하지 않는 파일은 정기적으로 정리됩니다. 기본적으로 정리 프로세스는 부팅 직후 실행되며, 이후에는 매일 실행됩니다. 임시 파일의 정리를 구성하는 방법에 대한 자세한 내용은 `tmpfiles.d(5)` 및 `systemd-tmpfiles(8)` man 페이지를 참조하세요.

/tmp 및 [/var/tmp](#) 경로는 밀접한 관련이 있으며 다양한 목적으로 존재합니다.

/run(런타임 데이터)

`/run` 디렉터리는 시스템 패키지에서 소량의 런타임 데이터(예: 소켓 파일)를 저장하는 데 사용됩니다. tmpfs 파일 시스템이며 권한이 있는 프로그램에서만 쓸 수 있습니다.

`/run/log` 디렉터리는 `/var/log`에 기록되기 전 또는 `/var/log` 파일 시스템을 사용할 수 있기 전에 시스템 구성 요소에서 로그를 저장하는 데 사용할 수 있습니다.

`/run/user/` 경로에는 사용자별 런타임 디렉터리가 포함되어 있습니다. 기본적으로 이는 사용자가 로그인할 때 systemd가 탑재하고 사용자가 더 이상 로그인하지 않을 때 지워지는 개별 tmpfs 파일 시스템입니다. [XDG 기본 디렉터리 사양](#)에 따라 이러한 경로는 `$XDG_RUNTIME_DIR` 환경 변수를 통해 직접 참조해서는 안 됩니다.

/usr(시스템 리소스)

`/usr` 계층 구조는 공급업체 제공 운영 체제 리소스를 위한 것입니다. [/usr/local](#) 계층 구조를 제외하고 OS 패키지 관리자를 제외하고 `/usr`의 어떤 것도 수정해서는 안 됩니다.

소프트웨어 애플리케이션은 `/usr`이 읽기 전용일 수 있다고 가정해야 합니다. 휘발성 데이터에 `/usr` 계층 구조를 사용해서는 안 됩니다. [/usr/local](#)을 제외하고 OS 패키지 관리자가 수행한 패키지 설치/제거 외부에서 추가 또는 변경된 데이터에 `/usr` 계층 구조를 사용해서는 안 됩니다. OS 패키지 관리자는 모든 `/usr` 계층 구조([/usr/local](#) 제외)가 동일한 탑재 지점이라고 가정할 수 있습니다.

OS 패키지 관리자 외부에 설치되는 소프트웨어는 향후 OS 패키지 관리자의 간접 호출을 방해할 수 있으므로 `/usr`에 데이터를 저장해서는 안 됩니다. [/usr/local](#) 계층 구조는 예외이며, OS 패키지 관리자 외부의 소프트웨어용으로 예약되어 있습니다.

/usr/bin(실행 파일)

표준 검색 `$PATH`에 나타나야 하고 쉘에서 간접 호출하는 데 유용한 실행 파일입니다. 쉘에서 간접 호출하는 데 유용하지 않은 대몬 및 실행 파일은 대신 `/usr/lib` 또는 `/usr/libexec`에 있습니다.

/usr/include(C/C++ 헤더)

`/usr/include` 디렉터리에는 일반적으로 `-devel` 접미사가 있는 패키지에 포함된 C 및 C++ 헤더 파일이 포함되어 있습니다.

/usr/lib 및 **/usr/lib64**(공유 라이브러리)

Amazon Linux 2023에서 `/usr/lib64` 경로는 64비트 공유 라이브러리와 아키텍처에 따라 달라지는 패키지 데이터에 사용됩니다. AL2023은 32비트 사용자 공간 지원과 함께 제공되지 않으므로 64비트 공유 라이브러리만 사용할 수 있습니다.

`/usr/lib` 경로는 모든 아키텍처와 호환되는 OS 패키지의 정적 데이터를 위한 것입니다. 여기에는 일반적으로 셸에서 간접 호출되지 않는 실행 파일이 포함될 수 있으며, 이는 `/usr/libexec`에서도 찾을 수 있습니다. 공유 라이브러리는 `/usr/lib` 대신 `/usr/lib64`에서 찾을 수 있습니다.

/usr/local(시스템 관리자 설치 소프트웨어)

Amazon Linux 2023에서는 시스템 관리자가 OS가 소유하지 않고 OS가 터치하지 않는 소프트웨어를 설치할 때 `/usr/local`을 사용할 수 있습니다. 기본 `/usr/local` 계층 구조는 `/` 계층 구조를 미러링합니다.

/usr/share(공유 리소스)

설명서, 글꼴 및 시간대 데이터와 같은 공유 리소스는 `/usr/share`에 있습니다. 다양한 사양에 따라 이 디렉터리에 데이터가 저장되는 위치와 형식을 정확하게 지정하는 것이 일반적입니다.

/usr/share/doc(설명서)

패키지와 함께 제공되는 설명서는 `/usr/share/doc`에 저장됩니다.

/var(영구 변수 시스템 데이터)

/var/cache(캐시)

[`/var/lib`](#)와 달리 `/var/cache`에서 데이터를 삭제해도 애플리케이션이 다른 소스에서 `/var/cache` 데이터를 다시 빌드할 수 있어야 하므로 데이터가 손실되지 않습니다.

/var/lib(영구 시스템 데이터)

`/var/lib` 디렉터리는 영구 시스템 데이터에 사용됩니다. 다양한 시스템 구성 요소는 해당 구성 요소에 대한 비공개 데이터를 여기에 배치합니다. [`/var/cache`](#)와 달리 `/var/lib`에서 데이터를 삭제하면 데이터가 손실됩니다.

예를 들어 PostgreSQL 데이터베이스 서버는 기본적으로 데이터베이스 데이터를 `/var/lib/pgsql`에 저장합니다. 이 데이터의 레이아웃 및 파일 형식은 PostgreSQL에 비공개이며, 삭제된 것처럼 영구 데이터의 모습을 가지며, 사용자에게 데이터 손실이 발생합니다.

/var/log(영구 로그)

이 디렉터리는 영구 로그를 저장하는 데 사용됩니다. 소프트웨어는 로그 파일을 `/var/log`로 직접 저장하지 않고 `syslog(3)` 또는 `sd_journal_print(3)` API 직접 호출을 사용하는 것이 좋습니다.

Note

AL2023에서 [systemd 저널 대체 rsyslog](#)는 기본 Amazon Linux 2 구성과 눈에 띄는 차이점이 있습니다.

`journalctl`을 사용하여 로그를 읽는 방법에 대한 자세한 내용은 [journalctl](#) 매뉴얼 페이지를 참조하세요.

많은 애플리케이션은 `/var/log`에 있는 로그 파일을 쓰고 때로는 교체하는 자체 메커니즘을 사용합니다. 로그 파일을 구성하는 방법은 이러한 애플리케이션에 대한 설명서를 참조하세요.

/var/spool(메일 및 프린터 대기열)

이 디렉터리는 메일 또는 프린터 대기열과 같은 영구 데이터에 사용됩니다.

/var/tmp(더 큰 임시 파일)

크기가 한정된 작은 임시 파일의 경우 [/tmp](#)를 대신 사용할 수 있습니다.

[/tmp](#)는 기본적으로 tmpfs 볼륨으로 구성되지만 `/var/tmp`는 기본적으로 루트 파일 시스템의 경로로 구성되므로 더 크고 영구적인 임시 파일의 위치에 해당합니다. 기본적으로 정기적으로 실행되는 정리 작업은 최근에 액세스하지 않은 파일을 제거합니다.

임시 파일의 정리를 구성하는 방법에 대한 자세한 내용은 `tmpfiles.d(5)` 및 `systemd-tmpfiles(8)` man man 페이지를 참조하세요.

[/tmp](#)와 마찬가지로 애플리케이션은 `$TMPDIR`보다 `/var/tmp` 환경 변수에 지정된 경로를 선호해야 합니다. 그런 다음 사용자는 `$TMPDIR` 환경 변수를 설정하여 애플리케이션이 `/var/tmp`에 사용해야 하는 경로를 재정의할 수 있습니다.

⚠ Warning

[/tmp](#)처럼 `/var/tmp`는 공유되므로 임시 파일을 생성하는 안전한 방법을 사용하는 것이 중요합니다. 자세한 내용은 [/tmp 및 /var/tmp의 안전한 사용](#)에 대한 업스트림 systemd 설명서를 참조하세요.

ℹ Note

[/tmp](#) 및 [/var/tmp](#)가 호스트 또는 다른 서비스와 공유되지 않는 샌드박스에서 서비스를 실행하는 `yes` 또는 `disconnected`로 설정된 `PrivateTmp=` 지시문으로 systemd 서비스를 구성하는 것이 가장 좋습니다.

동일한 프라이빗 임시 디렉터리를 공유하도록 두 서비스를 구성하는 방법을 비롯한 자세한 내용은 `systemd.exec(5)` man 페이지를 참조하세요.

[/tmp](#) 및 [/var/tmp](#) 경로는 밀접한 관련이 있으며 다양한 목적으로 존재합니다.

AL2023 업데이트

보안 업데이트와 새로운 기능을 활용하려면 AL2023 릴리스를 최신 상태로 유지하는 것이 중요합니다. AL2023을 사용하면 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#)를 통해 환경 전반에서 패키지 버전과 업데이트를 일치시킬 수 있습니다.

Warning

`dnf --releasever=latest update` 실행은 모범 사례가 아니며 프로덕션 환경에서 OS 업데이트를 처음 테스트하게 될 가능성이 높습니다.

`latest` 사용 대신 특정 AL2023 릴리스 버전을 사용합니다. 이렇게 하면 이전에 테스트한 것과 동일한 변경 사항을 프로덕션 인스턴스에 배포할 수 있습니다. 예를 들어 `dnf --releasever=2023.10.20260325 update`는 항상 2023.10.20260325 릴리스로 업데이트됩니다.

자세한 내용은 [AL2023 사용 설명서](#)의 [AL2023 업데이트](#) 섹션을 참조하세요.

주제

- [업데이트를 안전하게 배포하기 위한 모범 사례](#)
- [새 업데이트 알림 받기](#)
- [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#)
- [AL2023에서 패키지 및 운영 체제 업데이트 관리](#)
- [AL2023의 커널 라이브 패치](#)
- [AL2023에서 Linux 커널 업데이트](#)

업데이트를 안전하게 배포하기 위한 모범 사례

Amazon Linux 2023(AL2023)에는 운영 체제에 업데이트를 안전하게 배포하고 업데이트 간 변경 사항을 파악하고 필요한 경우 이전 버전으로 쉽게 되돌릴 수 있도록 설계된 여러 기능이 있습니다. 이 섹션에서는 10년이 넘는 내부 및 외부 Amazon Linux 사용 AWS 에서 얻은 교훈을 살펴봅니다.

Warning

`dnf --releasever=latest update` 실행은 모범 사례가 아니며 프로덕션 환경에서 OS 업데이트를 처음 테스트하게 될 가능성이 높습니다.

latest 사용 대신 특정 AL2023 릴리스 버전을 사용합니다. 이렇게 하면 이전에 테스트한 것과 동일한 변경 사항을 프로덕션 인스턴스에 배포할 수 있습니다. 예를 들어 `dnf --releasever=2023.10.20260325 update`는 항상 2023.10.20260325 릴리스로 업데이트됩니다.

자세한 내용은 [AL2023 사용 설명서](#)의 [AL2023 업데이트](#) 섹션을 참조하세요.

OS 업데이트의 배포 안전을 계획하지 않으면 애플리케이션/서비스와 OS 업데이트 간의 예상치 못한 부정적 상호 작용으로 인한 영향이 상당히 커질 수 있으며, 최악의 경우 전체 중단에 이를 수도 있습니다. 모든 소프트웨어 문제와 마찬가지로 문제가 조기에 감지될수록 최종 사용자에게 미칠 수 있는 영향이 줄어듭니다.

기본적으로 사실이 아닌 다음 두 가지를 믿는 함정에 빠지지 않는 것이 중요합니다.

1. OS 공급업체는 OS 업데이트에서 절대 실수를 하지 않습니다.
2. 사용자가 의존하는 OS의 특정 동작이나 인터페이스가 OS 공급업체가 의존 가능한 것으로 간주하는 동작 및 인터페이스와 일치합니다.

즉, OS 공급업체와 사용자 모두 업데이트에 문제가 있다는 데 동의할 것입니다.

좋은 의도에 의존하지 말고 배포 안전성에 모든 OS 업데이트가 포함되도록 시스템을 마련합니다.

프로덕션 환경에 배포하여 새 OS 업데이트를 테스트하는 것은 권장되지 않습니다. OS를 배포의 다른 부분으로 간주하고 프로덕션 환경의 다른 변경 사항에 적합하다고 생각되는 것과 동일한 배포 안전 메커니즘을 적용하는 것이 좋습니다.

프로덕션 시스템에 배포하기 전에 모든 OS 업데이트를 테스트하는 것이 가장 좋습니다. 배포할 때 적절한 모니터링과 결합된 단계적 롤아웃이 권장됩니다. 단계적 롤아웃을 사용하면 문제가 발생할 경우(즉시 발생하지 않더라도) 플릿의 일부로 영향이 제한되고 추가 조사 및 완화가 진행되는 동안 업데이트의 추가 배포를 중지할 수 있습니다.

OS 업데이트로 인한 부정적인 영향을 완화하는 것이 가장 우선시되는 경우가 많으며, 그 다음으로 문제가 어디에 있든 해결합니다. OS 업데이트 도입으로 부정적인 영향이 발생하는 경우 이전의 정상 작동하는 것으로 확인된 OS 버전으로 되돌릴 수 있는 기능은 강력한 도구입니다.

Amazon Linux 2023에는 OS(또는 개별 패키지) 버전에 대한 모든 변경을 재현 가능하게 보장하는 강력한 새 기능인 [버전 관리 리포지토리를 통한 결정적 업그레이드](#)가 도입되었습니다. 따라서 한 OS 버전에서 다음 버전으로 이동할 때 문제가 발생하면 문제 해결 방법을 찾는 동안 정상 작동하는 것으로 알려진 OS 버전을 유지하는 간단한 메커니즘을 이용할 수 있습니다.

AL2023에서는 새 패키지 업데이트를 릴리스할 때마다 고정할 수 있는 새 버전과 해당 버전에 고정된 새 AMI가 제공됩니다. [AL2023 릴리스 노트](#)는 각 릴리스의 변경 사항을 다루고 [AL2023용 Amazon Linux 보안 권고](#) 섹션은 패키지 업데이트에서 해결된 보안 문제를 다룹니다.

예를 들어 [2023.6.20241028](#) 릴리스에 있는 문제의 영향을 받은 경우 이전 릴리스인 [2023.6.20241010](#)의 AMI 및 컨테이너 이미지 사용으로 즉시 되돌릴 수 있습니다. 이 경우 후속 [2023.6.20241031](#) 릴리스에서 수정된 패키지 버그가 있었지만 [버전 관리 리포지토리를 통한 결정적 업그레이드](#)를 사용하면 영향을 받는 사람은 누구나 즉시 간단한 완화 조치를 취할 수 있습니다. 즉, 이전 이미지를 사용하기만 하면 됩니다.

[버전 관리 리포지토리를 통한 결정적 업그레이드](#)는 또한 현재 위치에서든 새 AMI 또는 컨테이너 이미지 실행을 통해서든 진행 중인 OS 업데이트 배포가 이후에 릴리스되는 OS 업데이트의 영향을 받지 않도록 보장합니다.

첫 번째 예제에서 플릿 A는 [2023.5.20241001](#)에서 [2023.6.20241010](#) 릴리스로 업데이트를 배포하는 도중에 [2023.6.20241028](#) 릴리스가 출시되는 대규모 플릿입니다. [버전 관리 리포지토리를 통한 결정적 업그레이드](#)는 플릿 A의 배포가 적용 중인 업데이트를 변경하지 않고 계속 진행됨을 의미합니다.

플릿의 1%에 처음 배포한 다음 100%에 도달할 때까지 5%, 10%, 20%, 40%와 같은 웨이브 기반 또는 단계 기반 배포 전략의 목적은 더 넓게 배포하기 전에 제한된 방식으로 변경 사항을 테스트할 수 있도록 하는 것입니다. 이러한 유형의 배포 전략은 일반적으로 프로덕션 변경 사항을 배포하는 모범 사례로 간주됩니다.

웨이브 기반 배포 전략과 플릿 A의 [2023.6.20241010](#) 업데이트가 한 번에 많은 호스트에 배포되는 단계에 있을 때 [2023.6.20241028](#)이 릴리스되었다는 사실은 [버전 관리 리포지토리를 통한 결정적 업그레이드](#) 사용 덕분에 진행 중인 배포에 영향을 미치지 않습니다.

플릿 B가 [2023.5.20240708](#) 같은 이전 버전을 실행 중이고 [2023.6.20241028](#)로 업데이트를 배포하기 시작했으며 플릿 B가 해당 버전의 문제에 영향을 받았다면 배포 초기에 이를 발견할 수 있을 것입니다. 이때 해당 문제에 대한 수정 사항을 사용할 수 있을 때까지 롤아웃을 일시 중지할지, 아니면 그 사이에 플릿 A가 실행 중인 동일한 버전인 [2023.6.20241010](#)의 배포를 시작하여 플릿 B가 [2023.5.20240708](#)과 [2023.6.20241010](#) 사이의 모든 업데이트를 받을 수 있도록 할지 결정할 수 있습니다.

OS 업데이트를 즉시 수행하지 않으면 문제가 발생할 수 있다는 점에 유의해야 합니다. 새 업데이트에는 환경과 관련이 있을 수 있는 버그 및 보안 수정이 포함될 수 있습니다. 자세한 내용은 [Amazon Linux 2023 보안 및 규정 준수](#) 및 [AL2023에서 패키지 및 운영 체제 업데이트 관리](#) 섹션을 참조하세요.

새 OS 업데이트를 쉽게 가져오고, 프로덕션에 배포하기 전에 테스트하고, 웨이브 기반 배포와 같은 메커니즘을 사용하여 부정적인 영향을 최소화할 수 있도록 배포 시스템을 구성하는 것이 중요합니다. OS 업데이트의 부정적인 영향을 완화하려면 배포 시스템이 이전의 정상 작동하는 것으로 알려진 OS

버전을 가리키도록 하는 방법을 알고 있어야 하며, 문제가 해결되면 더 이상 이전의 정상 버전에 고정되지 않고 새로운 정상 버전으로 이동해야 합니다.

마이너 업데이트 준비

AL2023의 새 포인트 릴리스와 같은 OS에 대한 소규모 업데이트를 준비하는 것은 노력이 전혀 필요하지 않도록 하기 위한 것입니다. 예정된 변경 사항은 [AL2023 릴리스 노트](#)를 참조하세요.

종료되는 [패키지의 지원 기간](#)에는 언어 런타임의 최신 버전으로 이동하는 작업이 포함될 수 있습니다 (예: [AL2023에서 PHP 사용](#) 사용). 지원 기간이 끝나기 전에 충분히 여유를 두고 새로운 언어 런타임 버전으로 이동하여 미리 준비하는 것이 좋습니다.

[pcre 버전 1](#) 같은 패키지의 경우 미리 계획을 세우고 코드를 대체품으로 마이그레이션할 수도 있습니다. 이 경우 대체품은 pcre 버전 2입니다. 예상치 못한 차질에 대비할 시간을 확보하기 위해 가능한 한 빨리 이렇게 하는 것이 가장 좋습니다.

[Berkeley DB\(libdb\)](#) 같이 직접적인 대체품이 없는 경우 사용 사례에 따라 선택해야 할 수 있습니다.

메이저 업데이트 준비

운영 체제의 새 메이저 버전으로 업데이트하는 것은 거의 보편적으로 계획, 변경되거나 더 이상 사용되지 않는 기능에 적응하기 위한 작업, 배포 전 테스트가 필요한 것으로 간주됩니다. 다음 메이저 버전으로 이동하기 전에 더 이상 사용되지 않거나 제거된 기능의 사용을 해결하는 등 Amazon Linux 2023의 다음 메이저 버전을 점진적으로 준비하는 것은 드문 일이 아닙니다.

예를 들어 AL2에서 AL2023으로 이동할 때 [AL2에서는 기능이 더 이상 사용되지 않고 AL2023에서는 제거됨](#) 섹션을 읽으면 AL2를 계속 사용하면서 AL2023을 준비할 수 있는 여러 안전하고 작은 단계를 수행할 수 있습니다. 예를 들어 모든 [Python 2.7은 Python 3으로 대체되었습니다](#). 사용(yum 패키지 관리자에서와 같은 OS 사용은 제외)을 [AL2023에서 Python 사용](#) 사용을 준비하기 위해 Python 3으로 마이그레이션할 수 있습니다. [PHP](#)를 사용하는 경우 (PHP 8.2 [AL2 Extra](#)를 통해) AL2와 AL2023 모두 PHP 8.2를 제공하므로 PHP 버전 마이그레이션과 OS 마이그레이션을 동시에 수행할 필요가 없습니다.

AL2023을 사용하는 동안 현재 Amazon Linux 2023의 다음 메이저 버전에 대비할 수도 있습니다.

[AL2023에서 사용 중단](#) 섹션에서는 AL2023에서 더 이상 사용되지 않으며 제거될 예정인 기능 및 패키지를 다룹니다.

예를 들어 init 스크립트와 같은 나머지 [System V init \(sysvinit\)](#) 사용을 해당 systemd 항목으로 마이그레이션하면 미래에 대비할 수 있을 뿐만 아니라 전체 systemd 기능 세트를 사용하여 서비스 모

니터링, 재시작 방법 및 여부, 필요한 다른 서비스, 적응해야 하는 리소스 또는 권한 제약 조건 등을 관리할 수 있습니다.

32비트 지원과 같은 기능의 경우 지원 중단이 여러 주요 OS 버전에 걸쳐 진행될 수 있습니다. 32비트의 경우 Amazon Linux 1(AL1)은 [32비트 x86\(i686\) AMI](#)를 더 이상 사용하지 않고, Amazon Linux 2는 [32비트 x86\(i686\) 패키지](#)를 더 이상 사용하지 않으며, Amazon Linux 2023은 [32비트 x86\(i686\) 런타임 지원](#)을 더 이상 사용하지 않습니다. [IMDSv1](#)에서의 전환도 여러 주요 OS 버전에 걸쳐 진행됩니다. 이러한 유형의 변경 사항의 경우 일부 고객이 적응하는 데 더 오래 걸린다는 점을 고려하여, Amazon Linux 2023에서 해당 기능을 더 이상 사용할 수 없게 되기 전까지 충분한 여유 시간이 제공됩니다.

더 이상 사용되지 않는 기능 목록은 OS 수명 동안 업데이트되며 변경 사항을 최신 상태로 유지하는 것이 좋습니다.

새 업데이트 알림 받기

새 AL2023 AMI가 릴리스될 때마다 알림을 받을 수 있습니다. 다음 주제에 대한 알림이 [Amazon SNS](#)에 게시됩니다.

```
arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates
```

새 AL2023 AMI가 게시되면 메시지가 여기에 게시됩니다. AMI 버전이 메시지에 포함됩니다.

이러한 메시지는 여러 방식으로 수신할 수 있습니다. 다음 방법을 사용하는 것이 좋습니다.

1. [Amazon SNS 콘솔](#)을 엽니다.
2. 필요한 경우 네비게이션 바에서 AWS 리전을 미국 동부(버지니아 북부)로 변경합니다. 구독 중인 SNS 알림을 만든 리전을 선택해야 합니다.
3. 탐색 창에서 구독과 구독 생성을 선택합니다.
4. 구독 생성 대화 상자에서 다음과 같이 수행합니다.
 - a. 주제 ARN의 경우, Amazon 리소스 이름(ARN)인 **arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates**를 복사하여 붙여 넣습니다.
 - b. 프로토콜에서 이메일을 선택합니다.
 - c. 엔드포인트에 알림 받을 이메일 주소를 입력합니다.
 - d. 구독 생성을 선택합니다.
5. "AWS 알림 - 구독 신청 확인"이라는 제목의 확인 이메일을 받게 됩니다. 이메일을 열고 구독 확인을 선택하여 구독 신청을 완료합니다.

AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드

Note

기본적으로 AL2023 인스턴스에는 시작 시 심각하고 중요한 추가 보안 업데이트가 자동으로 수신되지 않습니다. 처음에는 AL2023 버전 및 선택한 AMI에서 사용할 수 있었던 업데이트가 포함되어 있습니다.

메이저 및 마이너 릴리스에 포함된 업데이트를 관리하세요.

AL2023을 통해 환경 전반에서 패키지 버전과 업데이트 간의 일관성을 유지할 수 있습니다. 또한 동일한 Amazon Machine Image (AMI)의 여러 인스턴스를 일관성있게 유지할 수 있습니다. 기본으로 설정된 AL2023의 버전 지정 리포지토리 기능으로 결정적 업그레이드를 하면 특정 목적의 일정에 따라 업데이트할 수 있습니다.

새 패키지 업데이트를 릴리스할 때마다 사용할 새 버전과 해당 버전에서 사용할 새 AMI가 포함됩니다.

AL2023은 리포지토리의 특정 버전에서 사용 가능합니다. 이는 메이저 버전과 마이너 버전 모두에서 지원을 받습니다. SSM 파라미터에서 볼 수 있는 AL2023 AMI는 항상 최신 버전입니다. 여기에는 중요한 보안 업데이트를 포함하여 최신 패키지 및 업데이트가 있습니다.

기존 AMI에서 인스턴스를 시작하는 경우 자동으로 업데이트되지 않습니다. 프로비저닝 기능으로 설치된 추가 패키지는 기존 AMI의 리포지토리 버전에 매핑됩니다.

이 기능을 사용하면 환경 전반에서 패키지 버전과 업데이트 간의 일관성을 유지할 수 있습니다. 특히 같은 AMI에서 여러 인스턴스를 실행하는 경우 그렇습니다. 필요에 맞는 일정에 따라 업데이트할 수 있습니다. 특정 리포지토리 버전으로 고정할 수 있어서 실행 시 특정 업데이트 세트를 적용할 수도 있습니다.

메이저 버전과 마이너 버전 업그레이드의 차이점은 다음과 같습니다.

AL2023 메이저 버전 릴리스에는 대규모 업데이트가 포함되며 패키지를 추가, 삭제 또는 업데이트할 수 있습니다. 호환성을 유지하려면 인스턴스를 새 메이저 버전으로 업그레이드하기 전에 해당 버전에서 애플리케이션을 테스트해야 합니다.

AL2023 마이너 버전 릴리스에는 기능 및 보안 업데이트가 포함되지만 패키지 변경 사항은 포함되지 않습니다. 이렇게 하면 새 버전에서도 Linux 기능 및 시스템 라이브러리 API를 계속 사용할 수 있습니다. 업데이트하기 전에 애플리케이션을 테스트하지 않아도 됩니다.

업데이트를 사용할 수 있는 시기 알기

업데이트를 적용하려면 업데이트를 사용할 수 있는지를 확인한 다음 업데이트를 배포하는 방법을 알아야 합니다.

새 AL2023 AMI가 릴리스될 때 파생된 AMI를 빌드하기 위해 [EC2 Image Builder](#)는 AMI를 자동으로 빌드, 패치 및 테스트합니다. 자체 AMI 빌드 파이프라인을 트리거하거나 기본 AMI를 사용하기 위해 [새 업데이트 알림 받기](#)의 조치를 취할 수 있습니다.

현재 위치 패치의 경우 [AWS Systems Manager 패치 관리자](#)와 같은 도구를 사용하여 플릿 전체에서 적용 업데이트를 오케스트레이션할 수 있습니다.

다른 AL2023 기반 퍼블릭 AMI의 경우 해당 AMI의 공급자는 자체 릴리스 일정 및 알림 방법을 가질 수 있습니다. 파생 AMI 또는 컨테이너 이미지를 사용하는 경우 업데이트가 릴리스될 때 게시자의 설명서를 확인하세요.

각 릴리스의 변경 사항은 [AL2023 릴리스 노트](#)에 설명되어 있습니다. 보안 업데이트는 [Amazon Linux 보안 센터\(ALAS\)](#)에 게시되어 있습니다.

AL2023 리포지토리에서 사용할 수 있는 패키지 업데이트 관리

AL2023 리포지토리의 최신 버전을 릴리스해도 구 버전을 모두 사용할 수 있습니다. 기본적으로 리포지토리 버전 관리용 플러그인은 AMI 구축에 사용된 것과 동일한 버전입니다. 패키지 업데이트를 관리하려면 다음 단계를 따르세요.

1. 다음 명령을 실행하여 사용 가능한 리포지토리 버전을 검색합니다.

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

2. 다음 명령을 실행하여 버전을 선택하세요.

```
$ sudo dnf upgrade --releasever=version
```

이 명령으로 현재 Amazon Linux 릴리스 버전에서 명령줄에 지정된 릴리스 버전으로 업데이트할 수 있습니다. 패키지 업데이트 목록은 dnf에서 제공합니다. 업데이트하기 전에 업데이트를 확인해야 합니다. 업데이트가 완료되면 새 릴리스 버전은 향후 모든 dnf 활동의 기본 릴리스 버전이 됩니다.

자세한 내용은 [AL2023에서 패키지 및 운영 체제 업데이트 관리](#) 단원을 참조하십시오.

인스턴스 교체를 통한 결정적 업데이트

Amazon Linux 2023의 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#) 기능을 사용하면 인스턴스 교체를 통해 업데이트된 AL2023 버전을 결정적이고 안전하게 배포할 수 있습니다. 결정적 업데이트는 새 버전이 점진적으로 출시됨에 따라 문제가 발견되면 문제의 원인을 확인하면서 이전 AMI로 간단히 되돌릴 수 있음을 의미합니다.

현재 위치에 패치를 적용하는 대신 인스턴스 교체를 사용하면 새 용량을 시작하는 것이 명확한 A 및 B 상태의 잘 테스트된 코드 경로일 수 있으므로 업데이트가 더 결정적이고 예측 가능합니다. 배포가 시작되기 전에 CI/CD 시스템에서 각 이전 및 이후 상태를 잘 테스트할 수 있습니다.

현재 위치 패치를 수행할 때 업데이트를 적용하기 전과 후 사이에 중간 상태가 많으므로 모든 상태 조합을 테스트하기가 더 어렵습니다.

결정적 업데이트와 함께 인스턴스 교체를 사용하는 OS 업데이트 전략은 블루/그린, 웨이브 및 단계 기반 배포 모델에 적합합니다.

버전 관리 리포지토리를 통한 결정적 업그레이드

주제

- [결정적 업그레이드 시스템 사용](#)
- [결정적 업그레이드 시스템의 선택적 업데이트](#)
- [결정적 업그레이드와 함께 연속 재정의 사용](#)

결정적 업그레이드 시스템 사용

Note

패키지 관리자의 기본 동작이 AL2에서 변경되었습니다.

결정적 업그레이드는 광범위한 배포 전에 프로덕션 환경의 모든 변경 사항을 완전히 테스트할 수 있는 강력한 방법입니다. 각각의 새 AL2023 AMI는 특정 버전의 AL2023으로 잠깁니다. 이를 통해 특정 AMI를 시작할 때 설치된 OS 패키지 버전의 결정적 동작을 확인할 수 있습니다. 현재 위치 업데이트는 특정 릴리스 버전일 수 있으므로 플릿 전반에 걸쳐 결정적 동작을 보장합니다. 새 AMI 또는 현재 위치 업데이트 버전으로 이동할 때 CI/CD 파이프라인에서 각 버전을 테스트하여 프로덕션 환경에 배포하기 전에 잠재적 문제를 포착할 수 있습니다.

[AWS Systems Manager 패치 관리자](#) 같은 도구를 사용하여 플릿 전체에서 적용 업데이트를 오케스트레이션할 수 있습니다. 새 AL2023 AMI가 릴리스될 때 파생 AMIs 빌드하기 위해 [EC2 Image Builder](#)는 자동으로 AMI 빌드, 패치 및 테스트하거나 새 기본 AMI를 사용할 수 있는 시기를 알거나 자체 AMI 빌드 파이프라인을 트리거하기 위해 [새 업데이트 알림 받기](#)의 조치를 취할 수 있습니다.

특정 권고로 업데이트를 제한하는 방법에 대한 자세한 내용은 [현재 위치에서 보안 업데이트 적용](#) 섹션을 참조하세요.

현재 위치 패치의 경우 dnf 패키지 관리자를 사용할 수 있습니다. dnf upgrade 명령을 실행하면 시스템은 releasever 변수가 지정하는 저장소에서 업그레이드를 확인합니다. 유효한 releasever는 ## 버전이거나 **2023.10.20260325**와 같은 날짜 스탬프 버전입니다.

두 가지 방법 중 하나를 사용하여 releasever 값을 변경할 수 있습니다. 이러한 방법은 내림차순으로 시스템 우선 순위를 나열합니다. 즉, 방법 1은 방법 2와 3보다 우선하고 방법 2는 방법 3보다 우선합니다.

1. 명령줄 플래그를 사용했다면 그 값은 `--releasever=latest`입니다.
2. 오버라이드 변수 파일이 설정된 경우 지정된 값은 `/etc/dnf/vars/releasever`입니다.
3. 현재 설치된 system-release 패키지 버전입니다.

즉, 패키지 버전은 **2023.0.20230210**입니다.

```
$ rpm -q system-release
system-release-2023.0.20230210-0.amzn2023.noarch
```

새로 설치한 시스템에는 재정의 변수가 없습니다. 시스템이 설치된 system-release 버전을 제어하기 때문에 업그레이드가 불가능합니다.

```
$ 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 플래그로 원하는 버전을 지정하면 특정 버전의 패키지를 가져올 수 있습니다.

```
$ 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 옵션은 system-release와 /etc/dnf/vars/releasever 모두 재정의하므로 이 업그레이드의 결과는 다음과 같습니다.

1. 설치된 모든 패키지의 구 버전과 최신 버전 간의 변경 사항을 업그레이드합니다.
2. 업그레이드하면 시스템은 system-release 최신 버전의 리포지토리를 제어합니다.

항상 업데이트할 releasever(예: AL2023 릴리스)를 지정하면 플릿 전체에 결정적인 변경 사항이 적용됩니다. 버전 **A**를 시작하고 **B**로 업데이트한 다음 **C**로 업데이트했습니다.

결정적 업그레이드 시스템의 선택적 업데이트

Note

특정 업데이트를 선택하는 대신 새 릴리스의 모든 업데이트를 설치하는 것이 좋습니다. OS에 업데이트의 일부만 적용하는 것은 전체 업데이트를 수행하는 표준 관행에 대한 예외여야 합니다.

정식 릴리스 버전을 사용하는 시스템에 최신 릴리스에서 선택한 패키지를 설치하는 것이 좋습니다.

`dnf check-update`로 업그레이드할 패키지를 식별할 수 있습니다.

```
$ 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

업그레이드하고 싶은 패키지를 설치합니다. `sudo dnf upgrade --releasever=latest` 및 패키지 이름을 사용하면 `system-release` 패키지가 변경되지 않습니다.

```
$ 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`를 사용하면 `system-release`를 포함한 모든 패키지가 업데이트됩니다. 그러면 연속 재정의 설정하지 않는 한 최신 `system-release` 버전으로 유지됩니다.

결정적 업그레이드와 함께 연속 재정의의 사용

Note

결정적 업데이트를 통해 OS 변경 사항을 CI/CD 파이프라인에 통합할 수 있습니다. 결정적 업데이트를 비활성화하면 배포 전에 테스트할 수 없습니다.

--releasever=latest를 추가하는 대신 연속 재정의의 기능으로 변수 값을 ##으로 설정하여 시스템 잠금 해제를 할 수 있습니다. 항상 latest를 사용하면 AL2023의 동작이 AL2 업데이트 모델로 되돌아갑니다. 패키지 관리자에 대한 직접 호출은 항상 최신 릴리스를 보고, 특정 버전의 OS에 잠기지 않습니다.

Warning

결정적 업데이트의 지속적인 재정의의를 사용하여 패키지 관리자를 잠금 해제하면 애플리케이션과 프로덕션 OS 업데이트 간의 잠재적 비호환성을 발견할 위험이 있습니다.

비호환성은 드물지만 OS 업데이트를 통해 새 코드 변경 사항을 환경에 통합하는 경우 통합 테스트를 통해 프로덕션 환경에 부정적인 영향을 미치는 코드 변경 사항을 배포하지 못할 수 있습니다.

```
$ 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

재정의 변수 `/etc/dnf/vars/releasever`를 사용한 경우 다음 명령으로 재정의 값을 삭제하고 기본 잠금 동작을 복원할 수 있습니다.

```
$ sudo rm /etc/dnf/vars/releasever
```

특정 버전 대신 `latest`를 사용하기 위해 영구 재정의의 사용하는 것은 AL2의 기본 동작과 비슷합니다. AL2를 기반으로 AMI를 빌드하여 이 동작을 비활성화하고 AL2023에서 기본적으로 가져오는 것과 같은 특정 패키지 버전에 잠그는 서비스가 있습니다.

결정적 업데이트를 비활성화하는 대신 인스턴스를 새 AMI에서 시작된 인스턴스로 교체하는 것이 좋습니다. 인스턴스 교체가 옵션이 아닌 경우 [AWS Systems Manager 패치 관리자](#)와 같은 도구를 사용하여 플릿 전체에서 업데이트 적용을 오케스트레이션하는 것이 좋습니다. [EC2 Image Builder](#)는 AL2023 기본 이미지에서 파생된 자체 AMI를 자동으로 빌드, 패치 및 테스트할 수도 있습니다. 자체 AMI 빌드 파이프라인을 트리거하기 위해 [새 업데이트 알림 받기](#)의 조치를 취할 수도 있습니다.

사전 프로덕션 환경에서 `latest`를 사용한 다음 `latest`를 사용하여 프로덕션에 배포해도 OS 업데이트와 애플리케이션 간의 문제로부터 보호되지 않습니다. 새로운 AL2023 릴리스는 언제든지 사용할 수 있으므로 프로덕션 환경에서의 `latest` 사용 모두가 위험을 수반합니다.

AL2023에서 패키지 및 운영 체제 업데이트 관리

이전 Amazon Linux 버전과 달리, AL2023 AMI는 특정 Amazon Linux 리포지토리 버전에서만 사용할 수 있습니다. AL2023 인스턴스 보안 및 버그를 모두 수정하려면 DNF 구성을 사용 가능한 최신 릴리스 버전으로 업데이트하세요. 또는 최신 AL2023 인스턴스를 실행하세요.

이 섹션에서는 실행 중인 인스턴스에서 DNF 패키지 및 리포지토리를 관리하는 방법에 대해 설명합니다. 또한 시작 시 사용 가능한 최신 Amazon Linux 리포지토리를 활성화하도록 사용자 데이터 스크립트에서 DNF를 구성하는 방법도 설명합니다. 자세한 내용은 [DNF 명령 참조](#) 섹션을 참조하세요.

새 AL2023 릴리스에서 사용할 수 있는 모든 업데이트를 적용하는 것이 좋습니다. 보안 업데이트만 선택하거나 특정 업데이트만 선택하는 것은 규칙이 아닌 예외여야 합니다. 특정 인스턴스와 [보안 권고](#)와 관련된 목록은 [적용 가능한 권고 사항 나열](#) 섹션을 참조하세요. 특정 [권고](#)와 관련된 업데이트만 설치하는 방법에 대한 자세한 내용은 [현재 위치에서 보안 업데이트 적용](#) 섹션을 참조하세요.

Important

취약성을 보고하거나 AWS 클라우드 서비스 또는 오픈 소스 프로젝트와 관련된 보안 문제가 있는 경우 [취약성 보고 페이지](#)를 사용하여 AWS 보안에 문의하세요.

주제

- [사용 가능한 패키지 업데이트 확인](#)
- [DNF 및 리포지토리 버전에서 보안 업데이트](#)
- [\(보안\) 업데이트 후 자동 서비스 재시작](#)
- [보안 업데이트를 적용하려면 언제 재부팅해야 하나요?](#)
- [최신 리포지토리 버전이 활성화된 상태로 인스턴스 시작](#)
- [패키지 지원 정보 가져오기](#)
- [dnf check-release-update로 최신 리포지토리 버전 확인](#)
- [새 리포지토리 추가, 사용 또는 사용 중지](#)
- [cloud-init를 사용하여 리포지토리 추가](#)

사용 가능한 패키지 업데이트 확인

`dnf check-update` 명령을 사용하여 시스템 업데이트가 있는지 확인할 수 있습니다. AL2023의 경우 명령에 `--releasever=version-number` 옵션을 추가하는 것이 좋습니다.

이 옵션을 추가하면 DNF가 최신 저장소의 업데이트도 확인합니다. 예를 들어, `dnf check-update` 명령을 실행하면 반환된 최신 버전을 *version-number*의 값으로 사용합니다.

최신 버전의 리포지토리를 사용하도록 인스턴스를 업데이트하면 업데이트될 모든 패키지 목록이 출력에 포함됩니다.

Note

`dnf check-update` 명령에 플래그 옵션으로 릴리스 버전을 지정하지 않으면 현재 구성된 리포지토리 버전만 확인합니다. 즉, 최신 버전의 리포지토리에 있는 패키지는 검사되지 않습니다.

Updates in a specific version

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너를 시작한 경우 [2023.1.20230628](#) 릴리스에서 사용할 수 있는 업데이트를 살펴보겠습니다.

Note

이 예제에서는 [2023.0.20230315](#) 및 [2023.1.20230628](#) 릴리스를 사용하며, 이는 AL2023의 최신 릴리스가 아닙니다. 최신 보안 업데이트가 포함된 최신 릴리스는 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 `.0`은 특정 릴리스의 이미지 버전을 나타냅니다. 이 이미지 버전은 일반적으로 0입니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성하여 업데이트를 확인할 수 있습니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

이제 `dnf check-update` 명령은 [2023.1.20230628](#) 릴리스에서 사용 가능한 업데이트를 확인하는 데 사용됩니다.

Note

패키지 업데이트를 적용하는 것은 권한이 있는 작업입니다. 컨테이너에서 실행할 때는 일반적으로 권한 상승이 필요하지 않지만 Amazon EC2 인스턴스와 같은 컨테이너화되지 않은 환경에서 실행하는 경우 권한을 상승시키지 않고 업데이트를 확인할 수 있습니다.

```
$ 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
bash-5.2#
```

2023c-1.amzn2023.0.1

amazonlinux

system-release 패키지 버전은 `dnf check-update --releasever=2023.1.20230628` 명령에서 요청된 [2023.1.20230628](#) 릴리스인 `dnf upgrade` 명령이 업데이트할 릴리스를 보여줍니다.

Updates in the latest version

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너를 시작한 경우 latest AL2023 버전에서 사용할 수 있는 업데이트를 살펴보겠습니다. 작성 시점에 latest 릴리스는 [2023.5.20240708](#)이므로 이 예제에 나열된 업데이트는 해당 릴리스를 기준으로 합니다.

Note

이 예제에서는 [2023.4.20240319](#) 및 [2023.5.20240708](#) 릴리스를 사용하며, 후자는 작성 시점의 최신 릴리스입니다. 최신 릴리스에 대한 자세한 내용은 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 .1은 특정 릴리스의 이미지 버전을 나타냅니다. 이미지 버전은 일반적으로 0이지만 이 예제에서는 이미지 버전이 1인 릴리스를 사용합니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성하여 업데이트를 확인할 수 있습니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

이제 `dnf check-update` 명령은 latest 릴리스에서 사용 가능한 업데이트를 확인하는 데 사용됩니다. 이 업데이트는 작성 당시 [2023.5.20240708](#)이었습니다.

Note

패키지 업데이트를 적용하는 것은 권한이 있는 작업입니다. 컨테이너에서 실행할 때는 일반적으로 권한 상승이 필요하지 않지만 Amazon EC2 인스턴스와 같은 컨테이너화되지 않은 환경에서 실행하는 경우 권한을 상승시키지 않고 업데이트를 확인할 수 있습니다.

```
$ dnf --releasever=latest check-update
```

```
Amazon Linux 2023 repository
```

```
78 MB/s | 25 MB 00:00
```

```
Last metadata expiration check: 0:00:04 ago on Mon Jul 22 17:39:13 2024.
```

amazon-linux-repo-cdn.noarch	2023.5.20240708-1.amzn2023	amazonlinux
curl-minimal.x86_64	8.5.0-1.amzn2023.0.4	amazonlinux
dnf.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
dnf-data.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
expat.x86_64	2.5.0-1.amzn2023.0.4	amazonlinux
glibc.x86_64	2.34-52.amzn2023.0.10	amazonlinux
glibc-common.x86_64	2.34-52.amzn2023.0.10	amazonlinux
glibc-minimal-langpack.x86_64	2.34-52.amzn2023.0.10	amazonlinux
krb5-libs.x86_64	1.21-3.amzn2023.0.4	amazonlinux
libblkid.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libcurl-minimal.x86_64	8.5.0-1.amzn2023.0.4	amazonlinux
libmount.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libnghttp2.x86_64	1.59.0-3.amzn2023.0.1	amazonlinux
libsmartcols.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
libuuid.x86_64	2.37.4-1.amzn2023.0.4	amazonlinux
openssl-libs.x86_64	1:3.0.8-1.amzn2023.0.12	amazonlinux
python3.x86_64	3.9.16-1.amzn2023.0.8	amazonlinux
python3-dnf.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
python3-libs.x86_64	3.9.16-1.amzn2023.0.8	amazonlinux
system-release.noarch	2023.5.20240708-1.amzn2023	amazonlinux
yum.noarch	4.14.0-1.amzn2023.0.5	amazonlinux
bash-5.2#		

system-release 패키지 버전은 `dnf upgrade` 명령이 업데이트할 릴리스를 보여줍니다.

이 명령의 경우 새 패키지가 있다면 반환 코드는 100입니다. 새 패키지가 없다면 반환 코드는 0입니다. 또한 업데이트할 모든 패키지도 출력에 나열됩니다.

DNF 및 리포지토리 버전에서 보안 업데이트

최신 패키지 업데이트 및 보안 업데이트는 최신 저장소 버전에만 제공됩니다. AL2023 AMI 구 버전에서 실행한 인스턴스의 경우 보안 업데이트를 설치하기 전에 리포지토리 버전을 업데이트해야 합니다. 이 `dnf check-release-update` 명령에는 시스템에 설치된 모든 패키지를 최신 저장소 버전으로 업데이트하는 업데이트 명령이 포함되어 있습니다.

Note

`dnf check-update` 명령에 플래그 옵션으로 릴리스 버전을 지정하지 않으면 현재 구성된 리포지토리 버전만 확인합니다. 즉, 리포지토리의 이후 버전에 있는 설치된 패키지에 대한 업데이트는 적용되지 않습니다.

이 섹션에서는 개별 업데이트를 선택하거나 보안 업데이트로 표시된 업데이트만 적용하는 대신, 사용할 수 있는 모든 업데이트를 적용하는 권장 업그레이드 경로를 다룹니다. 모든 업데이트를 적용하면 기존 인스턴스가 업데이트된 AMI를 시작하는 것과 동일한 패키지 세트로 이동합니다. 이러한 일관성은 플랫 전체에서 패키지 버전의 차이를 줄입니다. 특정 업데이트의 적용에 대한 자세한 내용은 [현재 위치에서 보안 업데이트 적용](#) 섹션을 참조하세요.

Applying updates in a specific version

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너를 시작한 경우 [2023.1.20230628](#) 릴리스에서 사용할 수 있는 업데이트를 적용합니다.

Note

이 예제에서는 [2023.0.20230315](#) 및 [2023.1.20230628](#) 릴리스를 사용하며, 이는 AL2023의 최신 릴리스가 아닙니다. 최신 보안 업데이트가 포함된 최신 릴리스는 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 `.0`은 특정 릴리스의 이미지 버전을 나타냅니다. 이 이미지 버전은 일반적으로 0입니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성하여 업데이트를 적용할 수 있습니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

이제 `dnf upgrade` 명령은 [2023.1.20230628](#) 릴리스에 있는 모든 업데이트를 적용하는 데 사용됩니다.

Note

패키지 업데이트를 적용하는 것은 권한이 있는 작업입니다. 컨테이너에서 실행할 때는 일반적으로 권한 상승이 필요하지 않지만 Amazon EC2 인스턴스 같은 컨테이너화되지 않은 환경에서 실행하는 경우 root 사용자로 `dnf upgrade` 명령을 실행해야 합니다. 이 작업은 `sudo` 또는 `su` 명령을 사용하여 수행할 수 있습니다.

```
$ 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 패키지 버전은 `dnf upgrade --releasever=2023.1.20230628` 명령에서 요청된 [2023.1.20230628](#) 릴리스인 `dnf upgrade` 명령이 업데이트할 릴리스를 보여줍니다.

기본적으로 `dnf`는 업데이트를 적용할지 확인하라는 메시지를 표시합니다. `dnf`에 대한 `-y` 플래그를 사용하여 이 프롬프트를 우회할 수 있습니다. 이 예제의 경우 `dnf upgrade -y --releasever=2023.1.20230628` 명령은 업데이트를 적용하기 전에 확인을 요청하지 않습니다. 이는 스크립트 또는 기타 자동화 환경에서 유용합니다.

업데이트 적용 여부를 확인하면 `dnf`가 업데이트를 적용합니다.

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_64.rpm	2.1 MB/s	88 kB	00:00
(3/25): libcurl-minimal-8.0.1-1.amzn2023.0.1.x86_64.rpm	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.amzn2023.0.3.x86_64.rpm	1.3 MB/s	35 kB	00:00
(6/25): rpm-build-libs-4.16.1.3-12.amzn2023.0.6.x86_64.rpm	2.8 MB/s	90 kB	00:00
(7/25): rpm-libs-4.16.1.3-12.amzn2023.0.6.x86_64.rpm	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.rpm	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_64 15 MB/s | 421 kB      00:00
(13/25): openssl-libs-3.0.8-1.amzn2023.0.3.x86_64 35 MB/s | 2.2 MB      00:00
(14/25): libxml2-2.10.4-1.amzn2023.0.1.x86_64.rpm 14 MB/s | 706 kB      00:00
(15/25): curl-minimal-8.0.1-1.amzn2023.x86_64.rpm 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.rpm 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

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너를 시작한 경우 latest AL2023 버전에서 사용할 수 있는 업데이트를 적용합니다. 작성 시점에 latest 릴리스는 [2023.5.20240708](#)이므로 이 예제에 나열된 업데이트는 해당 릴리스를 기준으로 합니다.

Note

이 예제에서는 [2023.4.20240319](#) 및 [2023.5.20240708](#) 릴리스를 사용하며, 후자는 작성 시점의 최신 릴리스입니다. 최신 릴리스에 대한 자세한 내용은 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 .1은 특정 릴리스의 이미지 버전을 나타냅니다. 이미지 버전은 일반적으로 0이지만 이 예제에서는 이미지 버전이 1인 릴리스를 사용합니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성하여 업데이트를 적용할 수 있습니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

이제 `dnf upgrade` 명령은 latest 릴리스에서 사용 가능한 업데이트를 적용하는 데 사용됩니다. 이 업데이트는 작성 당시 [2023.5.20240708](#)이었습니다.

Note

패키지 업데이트를 적용하는 것은 권한이 있는 작업입니다. 컨테이너에서 실행할 때는 일반적으로 권한 상승이 필요하지 않지만 Amazon EC2 인스턴스 같은 컨테이너화되지 않은 환경에서 실행하는 경우 root 사용자로 `dnf upgrade` 명령을 실행해야 합니다. 이 작업은 `sudo` 또는 `su` 명령을 사용하여 수행할 수 있습니다.

기본적으로 `dnf`는 업데이트를 적용할지 확인하라는 메시지를 표시합니다. 이 예제에서는 `dnf`에 대한 `-y` 플래그를 사용하여 이 프롬프트를 우회합니다.

```
$ 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 업데이트를 확인하려면 다음 중 하나 이상을 수행하세요.

- `dnf check-update` 명령을 실행합니다. 이렇게 하면 잠긴 Amazon Linux 버전에서 적용되지 않은 업데이트가 있는지 확인할 수 있습니다. `system-release` 패키지만 업데이트한 경우 인스턴스가 잠긴 리포지토리 버전은 변경되지만 해당 버전에서 사용 가능한 업데이트는 적용되지 않아 업데이트가 표시될 수 있습니다.
- Amazon Linux 리포지토리 업데이트 SNS 주제(`arn:aws:sns:us-east-1:137112412989:amazon-linux-2023-ami-updates`)를 구독하세요. 자세한 설명은 Amazon Simple Notification Service 개발자 안내서에서 [Amazon SNS 주제 구독](#)을 참조하세요.
- [AL2023 릴리스 노트](#)를 정기적으로 확인하세요.
- [dnf check-release-update로 최신 리포지토리 버전 확인](#)의 조치를 통해 새 버전을 확인합니다.

Important

보안 업데이트가 포함된 AL2023의 새 버전이 자주 릴리스됩니다. 관련 보안 패치를 최신 상태로 유지해야 합니다.

(보안) 업데이트 후 자동 서비스 재시작

Amazon Linux는 이제 [smart-restart](#) 패키지와 함께 제공됩니다. Smart-restart는 `systems` 패키지 관리자를 사용하여 패키지를 설치하거나 삭제할 때마다 시스템 업데이트 시 `systemd` 서비스를 다시 시작합니다. 이는 `dnf (update|upgrade|downgrade)`가 실행될 때마다 발생합니다.

Smart-restart는 `dnf-utils`의 `needs-restarting` 패키지와 사용자 지정 거부 목록 메커니즘을 사용하여 재시작해야 할 서비스와 시스템 재부팅 권장 여부를 결정합니다. 시스템 재부팅이 권장되면 재부팅 힌트 마커 파일(`/run/smart-restart/reboot-hint-marker`)이 생성됩니다.

smart-restart(를) 설치하려면

(다른 패키지와 마찬가지로) 다음 DNF 명령을 실행합니다.

```
$ sudo dnf install smart-restart
```

설치 후 후속 트랜잭션은 `smart-restart` 로직을 트리거합니다.

거부 목록

Smart-restart는 특정 서비스의 재시작을 차단하도록 지시할 수 있습니다. 차단된 서비스는 재부팅이 필요한지 여부를 결정하는 데 영향을 주지 않습니다. 추가 서비스를 차단하려면 다음 예제와 같이 `/etc/smart-restart-conf.d/`에 접미사가 `-denylist`인 파일을 추가합니다.

```
$ cat /etc/smart-restart-conf.d/custom-denylist
# Some comments
myservice.service
```

Note

재부팅이 필요한지 여부를 결정할 때 모든 `*-denylist` 파일을 읽고 평가합니다.

사용자 지정 후크

거부 목록 작성 외에도 smart-restart는 서비스를 다시 시작하려고 시도하기 전과 후에 사용자 지정 스크립트를 실행하는 메커니즘을 제공합니다. 사용자 지정 스크립트를 사용하여 준비 단계를 수동으로 수행하거나 나머지 재시작 또는 완료된 재시작을 다른 구성 요소에 알릴 수 있습니다.

`/etc/smart-restart-conf.d/`에서 접미사가 `-pre-restart` 또는 `-post-restart`인 모든 스크립트가 실행됩니다. 순서가 중요한 경우 다음 예제와 같이 실행 순서를 확인할 수 있도록 모든 스크립트에 숫자 접두사를 붙입니다.

```
$ ls /etc/smart-restart-conf.d/*-pre-restart
001-my-script-pre-restart
002-some-other-script-pre-restart
```

보안 업데이트를 적용하려면 언제 재부팅해야 하나요?

경우에 따라 업데이트를 적용하려면 Amazon Linux를 재부팅해야 합니다.

- Linux 커널 패키지 업데이트는 최신 보안 업데이트가 적용된 새 커널을 활성화하기 위해 재부팅이 필요합니다. 커널 라이브 패치를 사용하면 제한된 기간 동안 보안 업데이트를 연기할 수 있습니다. 자세한 내용은 [AL2023의 커널 라이브 패치](#) 섹션을 참조하세요.
- EC2 Metal 인스턴스에서 Amazon Linux는 (Intel CPU용 `microcode_ctl` 패키지와 AMD CPU용 `amd-ucode-firmware` 패키지를 통해) 마이크로코드 업데이트를 제공합니다. 이러한 마이크로코드 업데이트는 후속 인스턴스 재부팅 시에만 활성화됩니다. 가상화된 EC2 인스턴스의 경우 기본 [AWS Nitro 시스템](#)이 마이크로코드 업데이트를 처리합니다.

- 실행 중인 일부 systemd 서비스는 전체 시스템을 다시 시작한 후에만 올바르게 작동합니다. smart-restart 메커니즘은 재부팅 힌트를 남겨 이러한 상황에 대해 알려줍니다. [\(보안\) 업데이트 후 자동 서비스 재시작](#)을(를) 참조하세요.

최신 리포지토리 버전이 활성화된 상태로 인스턴스 시작

사용자 데이터 스크립트에 DNF 명령을 추가하면 Amazon Linux AMI 실행 시 설치할 RPM 패키지를 선택할 수 있습니다. 다음은 사용자 데이터 스크립트로 user-data 스크립트로 실행한 모든 인스턴스에 동일한 패키지 업데이트가 설치되어 있는지 확인하는 예시입니다.

```
#!/bin/bash
dnf upgrade --releasever=2023.0.20230210
# Additional setup and install commands below
dnf install httpd php7.4 mysql80
```

이 스크립트는 슈퍼 유저(루트)로 실행해야 합니다. 다음 명령으로 실행하세요.

```
$ sudo sh -c "bash nameofscript.sh"
```

자세한 내용은 Amazon EC2 사용 설명서의 [사용자 데이터 및 셸 스크립트](#) 섹션을 참조하세요.

Note

사용자 데이터 스크립트 대신 최신 Amazon Linux AMI 버전 또는 Amazon Linux AMI 기반의 사용자 지정 AMI를 실행하세요. 최신 Amazon Linux AMI에 필요한 업데이트가 모두 설치되어 있으며 특정 리포지토리 버전에서 실행되도록 구성되어 있습니다.

패키지 지원 정보 가져오기

다양한 오픈 소스 소프트웨어 프로젝트가 AL2023에 통합되어 있습니다. 각 프로젝트는 Amazon Linux에서 개별 관리되며 출시 및 지원 종료 일정도 다릅니다. DNF supportinfo 플러그인이 제공하는 패키지 메타데이터를 통해 이러한 패키지에 대한 Amazon Linux 정보를 알 수 있습니다. 다음은 **dnf supportinfo** 명령이 glibc 패키지의 메타데이터를 반환하는 예시입니다.

```
$ 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.

```

패키지 지원 정보는 [AL2023 릴리스 노트](#)의 [지원 설명](#) 섹션에서도 확인할 수 있습니다.

dnf check-release-update로 최신 리포지토리 버전 확인

AL2023 인스턴스는 DNF 유틸리티를 사용하여 리포지토리를 관리하고 RPM 패키지 업데이트를 할 수 있습니다. Amazon Linux 리포지토리에서 이 패키지를 사용할 수 있습니다. DNF 명령 `dnf check-release-update`로 최신 DNF 리포지토리 버전을 확인할 수 있습니다.

Note

AL2023 컨테이너 이미지에는 기본적으로 `dnf check-release-update` 명령이 포함되지 않습니다.

```
$ 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)'`가 실행되면 `dnf`는 `check-release-update` 명령을 제공하는 패키지, 즉 `dnf-plugin-release-notification` 패키지를 설치합니다. 아래 예제에서는 무음 출력을 위해 `-q` 인수가 `dnf`에 제공됩니다.

```
$ dnf -y -q install 'dnf-command(check-release-update)'
```

```
Installed:
```

```
dnf-plugin-release-notification-1.2-1.amzn2023.0.2.noarch
```

Amazon EC2 인스턴스와 같은 컨테이너화되지 않은 환경에서는 기본적으로 `check-release-update` 명령이 포함됩니다.

```
$ 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
```

그러면 사용 가능한 모든 최신 DNF 리포지토리 버전을 알 수 있습니다. 결과가 나오지 않으면 DNF 이 현재 사용 가능한 최신 버전이라는 뜻입니다. 현재 설치된 system-release 패키지 버전은 releasever DNF 변수로 설정되어 있습니다. 다음 명령을 실행하여 현재 리포지토리 버전을 확인합니다.

```
$ rpm -q system-release --qf "%{VERSION}\n"
```

DNF 패키지 트랜잭션(예: 설치, 업데이트 또는 제거 명령)을 실행하면 최신 리포지토리 버전을 알리는 메시지가 나타납니다. 예를 들어, AL2023 구 버전으로 실행한 인스턴스에 httpd 패키지를 설치하면 다음과 같은 출력이 나타납니다.

```
$ 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!

새 리포지토리 추가, 사용 또는 사용 중지

Warning

AL2023과 함께 사용하도록 설계된 리포지토리만 추가합니다.

다른 배포용으로 설계된 리포지토리는 현재 작동할 수 있지만 AL2023의 패키지 업데이트 또는 AL2023과 함께 사용하도록 설계되지 않은 리포지토리에서도 계속 작동할 것이라는 보장은 없습니다.

기본 Amazon Linux 리포지토리와 다른 리포지토리에서 패키지를 설치하려면 리포지토리가 어디에 있는지 알 수 있도록 DNF 패키지 관리 시스템을 구성해야 합니다.

패키지 리포지토리에 대해 dnf에 알려려면 `/etc/yum.repos.d/` 디렉터리의 해당 리포지토리에 대한 구성 파일에 리포지토리 정보를 추가합니다. 많은 타사 리포지토리는 구성 파일 콘텐츠 또는 구성 파일이 포함된 설치 가능한 패키지를 제공합니다.

Note

리포지토리는 `/etc/dnf/dnf.conf` 파일에서 직접 구성할 수 있지만 권장되지 않습니다. 각 리포지토리는 `/etc/yum.repos.d/`의 자체 파일에 구성하는 것이 좋습니다.

다음 명령을 실행하여 현재 활성화된 리포지토리를 확인할 수 있습니다.

```
$ 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 옵션 플래그를 추가하지 않으면 Repo-id, Repo-name 및 Repo-status 정보만 출력에 포함됩니다.

/etc/yum.repos.d 디렉터리에 **yum** 리포지토리 추가하기

1. `.repo` 파일의 위치를 찾습니다. 이 예시에서 `.repo` 파일은 <https://www.example.com/repository.repo>에 있습니다.
2. `dnf config-manager` 명령을 사용하여 리포지토리를 추가합니다.

```
$ 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
```

리포지토리를 설치한 후 다음 절차에 따라 리포지토리를 활성화합니다.

`/etc/yum.repos.d`에서 yum 리포지토리를 사용하려면 `--enable` 플래그 및 `#####` 이름과 함께 `dnf config-manager` 명령을 사용합니다.

```
$ sudo dnf config-manager --enable repository
```

Note

리포지토리 사용 중지하려면 동일한 명령 구문을 사용하되 명령을 `--enable`에서 `--disable`로 변경하세요.

cloud-init를 사용하여 리포지토리 추가

이 방법으로 리포지토리를 추가할 수 있지만 `cloud-init` 프레임워크를 사용하여 새 리포지토리를 추가할 수도 있습니다.

새 패키지 리포지토리를 추가하려면 다음 템플릿을 사용하는 것이 좋습니다. 이 파일을 로컬에 저장하는 것을 추천합니다.

```
#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-init를 사용하면 좋은 점 한 가지는 구성 파일에 `packages:` 섹션을 추가할 수 있다는 것입니다. 설치하려는 패키지 이름을 이 섹션에 추가할 수 있습니다. 패키지는 기본 리포지토리 또는 cloud-config 파일에 추가한 새 리포지토리에서 설치할 수 있습니다. YAML 파일 구조에 대한 자세한 내용은 *cloud-init* 설명서의 [YUM 리포지토리 추가](#)(Adding a YUM repository)를 참조하세요.

YAML 형식 파일을 설정한 후에는 AWS CLI의 cloud-init 프레임워크에서 해당 파일을 실행할 수 있습니다. `--userdata` 옵션과 `.yaml` 파일 이름을 포함해야 원하는 작업을 호출할 수 있습니다.

```
$ 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
```

AL2023의 커널 라이브 패치

AL2023의 커널 라이브 패치를 사용하면 실행 중인 애플리케이션을 재부팅하거나 중단하지 않고 실행 중인 Linux 커널에 특정 보안 취약성 및 중요 버그 패치를 적용할 수 있습니다. 또한 커널 라이브 패치는 시스템을 재부팅할 수 있을 때까지 이러한 수정 사항을 적용하면서 애플리케이션의 가용성을 개선하는 데 도움이 될 수 있습니다.

AWS는 AL2023에 다음과 같은 두 가지 유형의 커널 라이브 패치를 릴리스합니다.

- 보안 업데이트 - Linux의 공통 보안 취약성 및 노출(Common Vulnerabilities and Exposures) 목록 업데이트가 있습니다. 이러한 업데이트는 일반적으로 Amazon Linux 보안 권고 등급을 사용하여 중요 또는 매우 중요로 등급이 매겨집니다. 대개 공통 취약점 등급 시스템(Common Vulnerability Scoring System) 점수 7 이상에 매핑됩니다. 경우에 따라 CVE가 할당되기 전에 AWS에서 업데이트를 제공할 수 있습니다. 이러한 경우 버그 수정 패치를 배포합니다.
- 버그 수정 - CVE와 관련없는 중요한 버그 및 안정성 문제에 대한 수정 사항이 포함되어 있습니다.

AWS는 릴리스 후 최대 3개월 동안 AL2023 커널 버전에 대한 커널 라이브 패치를 제공합니다. 3개월 후 커널 라이브 패치를 계속 받으려면 최신 커널 버전으로 업데이트해야 합니다.

AL2023 커널 라이브 패치는 기존 AL2023 리포지토리에서 서명된 RPM 패키지로 사용할 수 있으며, 패치는 기존 DNF 패키지 관리자 워크플로우를 사용하여 개별 인스턴스에 설치할 수 있습니다. 또는 AWS Systems Manager를 사용하여 관리형 인스턴스 그룹에 설치할 수 있습니다.

AL2023 커널 라이브 패치는 무료로 제공됩니다.

주제

- [제한 사항](#)
- [지원되는 구성 및 필수 조건](#)
- [커널 라이브 패치 작업](#)

제한 사항

커널 라이브 패치를 적용하는 동안 최대 절전 모드를 실행하거나, 고급 디버깅 도구(예: SystemTap, kprobes 및 eBPF 기반 도구)를 사용하거나, 커널 라이브 패치 인프라에 사용되는 ftrace 출력 파일에 액세스할 수 없습니다.

Note

기술적 제한으로 인해 일부 문제는 라이브 패치로 해결할 수 없습니다. 따라서 이러한 수정 사항은 커널 라이브 패치 패키지로 배송되지 않고 네이티브 커널 패키지 업데이트로만 배송됩니다. 네이티브 커널 패키지를 설치하고 시스템을 [업데이트 및 재부팅](#)하여 평소와 같이 패치를 활성화할 수 있습니다.

지원되는 구성 및 필수 조건

커널 라이브 패치는 AL2023을 실행하는 Amazon EC2 인스턴스 및 온프레미스 가상 머신에서 사용할 수 있습니다.

AL2023 커널 라이브 패치를 사용하려면 다음 사양이 필요합니다.

- 64비트 x86_64 또는 ARM64 아키텍처
- 커널 버전 6.1 또는 6.12

정책 요구 사항

AL2023 리포지토리에서 패키지를 다운로드하려면 Amazon EC2가 서비스 소유 Amazon S3 버킷에 액세스할 수 있어야 합니다. 환경에서 Amazon S3용 Amazon 가상 프라이빗 클라우드(VPC) 엔드포인트를 사용하는 경우 VPC 엔드포인트 정책이 해당 퍼블릭 버킷에 대한 액세스를 허용하는지 확인해야 합니다. 다음 표에서는 커널 라이브 패치를 위해 EC2가 액세스해야 할 수 있는 각 Amazon S3 버킷에 대해 설명합니다.

S3 버킷 ARN	설명
arn:aws:s3:::al2023-repos- <i>region</i> -de612dc2/*	AL2023 리포지토리가 포함된 Amazon S3 버킷

커널 라이브 패치 작업

인스턴스 자체 명령줄을 사용하여 개별 인스턴스에서 커널 라이브 패치를 활성화하고 사용할 수 있습니다. 또는 AWS Systems Manager를 사용하여 관리형 인스턴스 그룹에서 커널 라이브 패치를 활성화하고 사용할 수 있습니다.

다음 섹션에서는 명령줄을 사용하여 개별 인스턴스에서 커널 라이브 패치를 활성화하고 사용하는 방법에 대해 설명합니다.

관리형 인스턴스 그룹에서 커널 라이브 패치를 활성화하고 사용하는 방법에 대한 자세한 내용은 AWS Systems Manager 사용 설명서에서 [AL2023 인스턴스에서 커널 라이브 패치 사용](#) 섹션을 참조하세요.

주제

- [커널 라이브 패치 활성화](#)
- [사용 가능한 커널 라이브 패치 보기](#)
- [커널 라이브 패치 적용](#)
- [설치된 커널 라이브 패치 보기](#)
- [커널 라이브 패치 비활성화](#)

커널 라이브 패치 활성화

커널 라이브 패치는 AL2023에서 기본적으로 비활성화되어 있습니다. 라이브 패치를 사용하려면 커널 라이브 패치 DNF 플러그인을 설치하고 라이브 패치 기능을 활성화해야 합니다.

커널 라이브 패치 활성화

1. 커널 라이브 패치는 커널 버전 6.1가 포함된 AL2023에 사용할 수 있습니다. 커널 버전을 확인하려면 다음 명령을 실행합니다.

```
$ sudo dnf list kernel
```

2. 커널 라이브 패치용 DNF 플러그인을 설치합니다.

```
$ sudo dnf install -y kpatch-dnf
```

3. 커널 라이브 패치용 DNF 플러그인을 활성화합니다.

```
$ sudo dnf kernel-livepatch -y auto
```

또한 이 명령으로 구성된 리포지토리에서 최신 버전의 커널 라이브 패치 RPM을 설치할 수 있습니다.

4. 커널 라이브 패치용 DNF 플러그인이 성공적으로 설치되었는지 확인하려면 다음 명령을 실행합니다.

커널 라이브 패치를 활성화하면 빈 커널 라이브 패치 RPM이 적용됩니다. 커널 라이브 패치가 성공적으로 활성화된 경우 이 명령은 초기 빈 커널 라이브 패치 RPM(및 라이브 패치가 포함된 DNF 리포지토리를 설정하는 또 다른 RPM)을 포함하는 목록을 반환합니다.

```
$ 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 패키지를 설치합니다.

```
$ sudo dnf install -y kpatch-runtime
```

6. kpatch 서비스가 이전에 설치된 경우 업데이트합니다.

```
$ sudo dnf upgrade kpatch-runtime
```

7. kpatch 서비스를 시작합니다. 이 서비스로 초기화 또는 부팅 시 모든 커널 라이브 패치를 로드할 수 있습니다.

```
$ sudo systemctl enable kpatch.service && sudo systemctl start kpatch.service
```

사용 가능한 커널 라이브 패치 보기

Amazon Linux 보안 알림이 Amazon Linux 보안 센터에 게시됩니다. 커널 라이브 패치 알림을 비롯한 AL 2023 보안 알림에 대한 자세한 내용은 [Amazon Linux 보안 센터](#)를 참조하세요. 커널 라이브 패치 앞에 ALASLIVEPATCH가 붙습니다. Amazon Linux 보안 센터에는 버그를 해결하는 커널 라이브 패치가 나열되지 않을 수 있습니다.

명령줄을 사용하여 권고 및 CVE와 관련된 사용 가능한 커널 라이브 패치를 찾을 수도 있습니다.

권고와 관련된 사용 가능한 모든 커널 라이브 패치를 나열하려면

다음 명령을 사용합니다.

```
$ 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
```

CVE와 관련된 사용 가능한 모든 커널 라이브 패치를 나열하려면

다음 명령을 사용합니다.

```
$ 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
```

커널 라이브 패치 적용

DNF 패키지 관리자로 정기적인 업데이트를 하는 것과 같은 방식으로 커널 라이브 패치를 설치합니다. 커널 라이브 패치용 DNF 플러그인은 적용 가능한 커널 라이브 패치를 관리합니다.

Tip

커널 라이브 패치를 사용하여 커널을 정기적으로 업데이트하여 시스템을 재부팅할 수 있을 때 까지 중요한 보안 수정 사항과 긴급 보안 수정 사항을 수신하도록 하는 것이 좋습니다. 또한 라

이브 패치로 배포할 수 없는 네이티브 커널 패키지에 대한 추가 수정 사항을 사용할 수 있는지 확인하고 이러한 경우 커널 업데이트로 [업데이트하고 재부팅](#)하세요.

특정 커널 라이브 패치를 설치하거나 정기 보안 업데이트로 사용 가능한 커널 라이브 패치를 설치할 수 있습니다.

특정 커널 라이브 패치를 적용하려면

1. [사용 가능한 커널 라이브 패치 보기](#)에 설명된 명령 중 하나를 사용하여 커널 라이브 패치 버전을 설치합니다.
2. AL2023 커널에 커널 라이브 패치를 설치합니다.

```
$ sudo dnf install kernel-livepatch-kernel_version-package_version.amzn2023.x86_64
```

예를 들어, 다음 명령은 AL2023 커널 6.1.12-17.42 버전에 커널 라이브 패치를 설치합니다.

```
$ sudo dnf install kernel-livepatch-6.1.12-17.42-1.0-4.amzn2023.x86_64
```

정기 보안 업데이트와 함께 사용 가능한 커널 라이브 패치를 적용하려면

다음 명령을 사용합니다.

```
$ sudo dnf upgrade --security
```

버그 수정을 포함하려면 --security 옵션을 생략하세요.

Important

- 커널 버전은 커널 라이브 패치를 설치한 후에 업데이트되지 않으며 인스턴스를 재부팅한 후에만 새 버전으로 업데이트됩니다.
- AL2023 커널은 3개월 동안 커널 라이브 패치를 받습니다. 3개월이 경과한 후에는 해당 커널 버전에 대한 새로운 커널 라이브 패치가 릴리스되지 않습니다.
- 3개월 후에도 계속 커널 라이브 패치를 받으려면 인스턴스를 재부팅하여 최신 커널 버전으로 업데이트해야 합니다. 업데이트 후 다음 3개월 동안 커널 라이브 패치가 인스턴스에 설치됩니다.
- 커널 버전에 대한 지원 기간을 확인하려면 다음 명령을 실행합니다.


```
$ 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.

설치된 커널 라이브 패치 보기

설치된 커널 라이브 패치를 보려면

다음 명령을 사용합니다.

```
$ 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)

이 명령은 로드되고 설치된 보안 업데이트 커널 라이브 패치 목록을 반환합니다. 다음은 예시 출력입니다.

Note

단일 커널 라이브 패치에 여러 개의 라이브 패치가 포함되어 설치될 수 있습니다.

커널 라이브 패치 비활성화

커널 라이브 패치를 더 이상 사용할 필요가 없는 경우 언제든지 비활성화할 수 있습니다.

- livepatches 사용 비활성화:

- 플러그인 비활성화:

```
$ sudo dnf kernel-livepatch manual
```

- kpatch 서비스 비활성화:

```
$ sudo systemctl disable --now kpatch.service
```

- livepatch 도구 완전 삭제:

1. 플러그인 삭제:

```
$ sudo dnf remove kpatch-dnf
```

2. kpatch-runtime 삭제:

```
$ sudo dnf remove kpatch-runtime
```

3. 설치된 livepatches 삭제:

```
$ sudo dnf remove kernel-livepatch\*
```

AL2023에서 Linux 커널 업데이트

주제

- [AL2023의 Linux 커널 버전](#)
- [AL2023을 최신 커널 버전으로 업데이트](#)
- [AL2023 커널 - 자주 묻는 질문](#)

AL2023의 Linux 커널 버전

AL2023에는 Linux 커널의 장기 지원(LTS) 버전을 기반으로 하는 새 커널 버전이 정기적으로 포함됩니다.

AL2023은 원래 커널 6.1과 함께 2023년 3월에 릴리스되었습니다.

2025년 4월에 AL2023은 Linux 커널 6.12에 대한 지원을 추가했습니다. 이 커널에는 EEVDF 예약, FUSE 패스스루 I/O 지원, 새로운 Futex API, eBPF 개선 등 새로운 기능이 추가되었습니다. 또한 커널 6.12를 사용하면 사용자 공간 프로그램이 사용자 공간 새도우 스택 및 메모리 밀봉을 사용하여 런타임 시 자체 보안을 유지할 수 있습니다.

2026년 3월에 AL2023은 Linux 커널 6.18에 대한 지원을 추가했습니다. 업데이트된 커널 6.18은 프로세서 지원, 가상화, 보안 및 성능을 추가로 개선했습니다. 주목할 만한 기능에는 아키텍처 전반의 향상된 IOMMU 기능과 CPU 취약성 완화 관리를 위한 공격 벡터 제어가 포함됩니다. 성능 향상은 더 빠른 FSCRYPT 작업, 메모리 관리 개선, 새로운 옵트인 CPU별 배열 기반 캐싱 계층으로 Sheaves 도입을 통한 암호화 최적화를 통해 이루어집니다.

AL2023을 최신 커널 버전으로 업데이트

2026년 6월부터 AL2023은 기본 커널을 매년 업데이트합니다. AMIs [al2023-ami-kernel-default](#) 세트는 최신 LTS 커널로 업데이트되므로 새로 시작된 인스턴스에 새 커널 버전이 자동으로 제공됩니다. 이는 최신 보안 수정 및 성능 개선 사항을 최신 상태로 유지하는 가장 간단한 방법입니다.

특정 커널 버전을 선택하려면 원하는 커널이 사전 설치된 AMI를 선택하거나 기존 AL2023 EC2 인스턴스를 업그레이드하여 커널 6.12 또는 6.18로 AL2023을 실행할 수 있습니다.

특정 커널 버전으로 AL2023 AMI 실행

AWS 콘솔을 통해 사전 설치된 특정 커널을 사용하거나 SSM에서 특정 파라미터를 쿼리하여 AL2023 AMI를 실행하도록 선택할 수 있습니다. 쿼리할 SSM 키는 `/aws/service/ami-amazon-linux-latest/`로 시작하고 다음 중 하나가 뒤따릅니다.

커널 6.12의 경우

- arm64 아키텍처 `al2023-ami-kernel-6.12-arm64`
- arm64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-6.12-arm64`
- x86_64 아키텍처 `al2023-ami-kernel-6.12-x86_64`
- x86_64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-6.12-x86_64`

커널 6.18의 경우

- arm64 아키텍처 `al2023-ami-kernel-6.18-arm64`
- arm64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-6.18-arm64`
- x86_64 아키텍처 `al2023-ami-kernel-6.18-x86_64`
- x86_64 아키텍처 (미니멀 AMI) `al2023-ami-minimal-kernel-6.18-x86_64`

AL2023 AMI 선택에 대한 자세한 내용은 [SSM 파라미터 및를 사용하여 AL2023 시작 AWS CLI](#) 섹션을 참조하세요.

AL2023 인스턴스를 최신 커널로 업데이트

다음 단계에 따라 실행 중인 AL2023 인스턴스를 커널 6.12 또는 6.18로 인플레이스 업그레이드할 수 있습니다.

1. 현재 커널을 감지하고 대상 버전을 설정합니다.

```
# 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}"
```

- 대상 커널 패키지를 설치합니다.

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

- 대상 커널 패키지의 최신 버전을 가져옵니다.

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |
    sort -V | tail -1)
```

- 새 커널을 기본 커널로 설정합니다.

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

- 시스템을 재부팅합니다.

```
$ sudo reboot
```

- 이전 커널을 제거합니다.

```
# 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. 추가 커널 패키지를 대상 커널에 상응하는 패키지로 바꿉니다.

```
# Set your target version (change this to your desired kernel: 6.12 or 6.18)
$ TARGET_VERSION="6.12"
```

```
$ declare -A pkgs
$ pkgs=(
  [bpftool${SOURCE_VERSION}]=bpftool${TARGET_VERSION}
  [kernel${SOURCE_VERSION}-debuginfo]=kernel${TARGET_VERSION}-debuginfo
  [kernel${SOURCE_VERSION}-debuginfo-common]=kernel${TARGET_VERSION}-debuginfo-common
  [kernel${SOURCE_VERSION}-headers]=kernel${TARGET_VERSION}-headers
  [kernel${SOURCE_VERSION}-libbpf]=kernel${TARGET_VERSION}-libbpf
  [kernel${SOURCE_VERSION}-libbpf-devel]=kernel${TARGET_VERSION}-libbpf-devel
  [kernel${SOURCE_VERSION}-libbpf-static]=kernel${TARGET_VERSION}-libbpf-static
  [kernel${SOURCE_VERSION}-modules-extra-common]=kernel${TARGET_VERSION}-modules-extra-common
  [kernel${SOURCE_VERSION}-tools]=kernel${TARGET_VERSION}-tools
  [kernel${SOURCE_VERSION}-tools-devel]=kernel${TARGET_VERSION}-tools-devel
  [perf${SOURCE_VERSION}]=perf${TARGET_VERSION}
  [python3-perf${SOURCE_VERSION}]=python3-perf${TARGET_VERSION}
)
$ for pkg in "${!pkgs[@]}"; do
  rpm -q $pkg && sudo dnf -y swap $pkg "${pkgs["$pkg"]}" ;
done
```

8. (선택 사항) 이전 커널 버전의 커널 개발 제거:

```
$ rpm -q kernel${SOURCE_VERSION}-devel && sudo dnf remove -y kernel
${SOURCE_VERSION}-devel
```

이전 커널 버전으로 다운그레이드

언제든지 이전 커널 버전으로 다운그레이드해야 하는 경우 다음 단계를 사용합니다.

1. 현재 커널을 감지하고 대상 버전을 설정합니다.

```
# 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. 추가 커널 패키지를 대상 커널에 상응하는 패키지로 바꿉니다.

```
$ declare -A pkgs
$ pkgs=(
    [bpftool${TARGET_VERSION}]=bpftool${SOURCE_VERSION}
    [kernel${TARGET_VERSION}-debuginfo]=kernel${SOURCE_VERSION}-debuginfo
    [kernel${TARGET_VERSION}-debuginfo-common]=kernel${SOURCE_VERSION}-debuginfo-common
    [kernel${TARGET_VERSION}-headers]=kernel${SOURCE_VERSION}-headers
    [kernel${TARGET_VERSION}-libbpf]=kernel${SOURCE_VERSION}-libbpf
    [kernel${TARGET_VERSION}-libbpf-devel]=kernel${SOURCE_VERSION}-libbpf-devel
    [kernel${TARGET_VERSION}-libbpf-static]=kernel${SOURCE_VERSION}-libbpf-static
    [kernel${TARGET_VERSION}-modules-extra-common]=kernel${SOURCE_VERSION}-modules-extra-common
    [kernel${TARGET_VERSION}-tools]=kernel${SOURCE_VERSION}-tools
    [kernel${TARGET_VERSION}-tools-devel]=kernel${SOURCE_VERSION}-tools-devel
    [perf${TARGET_VERSION}]=perf${SOURCE_VERSION}
    [python3-perf${TARGET_VERSION}]=python3-perf${SOURCE_VERSION}
)
$ for pkg in "${!pkgs[@]}"; do
    rpm -q "${pkgs["$pkg"]}" && sudo dnf -y swap "${pkgs["$pkg"]}" $pkg ;
done
```

3. 대상 커널 패키지를 설치합니다.

```
$ sudo dnf install -y kernel${TARGET_VERSION}
```

4. 대상 커널 패키지의 최신 버전을 가져옵니다.

```
$ version=$(rpm -q --qf '%{version}-%{release}.%{arch}\n' kernel${TARGET_VERSION} |  
sort -V | tail -1)
```

5. 대상 커널을 기본 커널로 설정합니다.

```
$ sudo grubby --set-default "/boot/vmlinuz-$version"
```

6. 시스템을 재부팅합니다.

```
$ sudo reboot
```

7. 소스 커널을 제거합니다.

```
# 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 커널 - 자주 묻는 질문

1. 커널 업데이트 후 재부팅해야 하나요?

실행 중인 커널을 변경할 때마다 재부팅해야 합니다.

2. 여러 인스턴스에서 커널을 최신 상태로 유지하려면 어떻게 해야 하나요?

Amazon Linux는 인스턴스 플릿을 관리하는 기능을 제공하지 않습니다. [AWS Systems Manager](#)와 같은 도구를 사용하여 대규모 플릿을 패치하는 것이 좋습니다.

3. 현재 실행 중인 커널 버전을 확인하려면 어떻게 해야 하나요?

AL2023 인스턴스에서 다음 명령을 실행합니다.

```
$ uname -r
```

4. AL2023은 어떤 커널을 사용하도록 권장하나요?

다른 모든 AL2023 커널은 여전히 지원되지만 최신 AL2023 커널 6.18로 업그레이드하는 것이 좋습니다. 고객은 업그레이드하기 전에 워크로드를 테스트하는 것이 좋습니다.

5. 기존 애플리케이션이 AL2023 커널에서 작동하나요?

AL2023은 커널 6.1과 동일한 방식으로 최신 커널(6.12 또는 6.18)을 지원합니다. 애플리케이션이 작동하며 후드에서 개선이 진행되고 있습니다. 고객은 어떤 경우에도 최신 커널로 전환하기 전에 특정 워크로드를 테스트해야 합니다.

6. 커널 6.12 또는 6.18용 커널 헤더, 개발 패키지 및 추가 모듈을 설치하려면 어떻게 해야 합니까?

다음을 실행하세요.

```
$ 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. 커널 6.12 및 6.18은 얼마나 오래 지원되나요?

커널 6.12 및 6.18은 Amazon Linux 2023의 계획된 수명 종료인 2029-06-30까지 지원됩니다.

AL2023에서 프로그래밍 런타임 시작하기

AL2023은 일부 언어 런타임의 다양한 버전을 제공합니다. 당사는 동시에 여러 버전을 지원하는 업스트림 프로젝트와 작업합니다. 패키지를 검색하고 설치하는 `dnf` 명령을 사용하여 이름 버전이 지정된 패키지를 설치하고 관리하는 방법에 대한 정보를 찾아보세요.

다음 항목에서는 AL2023에서 각 언어 에코시스템이 어떻게 존재하는지 간략하게 설명합니다.

주제

- [AL2023의 C, C++, 및 Fortran](#)
- [AL2023에서 .NET 사용](#)
- [AL2023에서 Go 사용](#)
- [AL2023에 설치된 Java](#)
- [AL2023에서 Node.js 사용](#)
- [AL2023에 설치된 Perl](#)
- [AL2023에서 PHP 사용](#)
- [AL2023에서 Python 사용](#)
- [AL2023에서 Ruby 사용](#)
- [AL2023에서 Rust 사용](#)
- [AL2023에서 Swift 사용](#)
- [TypeScriptAL2023에서 사용](#)

AL2023의 C, C++, 및 Fortran

AL2023에는 GNU 컴파일러 컬렉션(GCC)과 저사양 가상 머신(LLVM) Clang 프론트엔드 모두 포함되어 있습니다.

GCC 주요 버전은 AL2023이 지원 종료되기 전까지 계속 사용할 수 있습니다. 마이너 릴리스에 버그 수정이 포함되어 있으며 이는 AL2023 릴리스에 포함될 수 있습니다. 기타 버그, 성능 및 보안 수정 사항은 AL2023에 설치된 GCC의 주요 버전으로 백포트될 수 있습니다.

AL2023에는 C(gcc), C++(g++) 및 Fortran(gfortran) 프론트엔드가 있는 기본 컴파일러로 GCC 버전 11이 포함되어 있습니다. 또한 AL2023은 기본 버전과 함께 설치할 수 있는 선택적 대체 컴파일러로 GCC 버전 14를 제공합니다.

AL2023에서 Ada(gnat), Go(gcc-go), Objective-C 또는 Objective-C++ 프론트엔드를 활성화할 수 없습니다.

AL2023 RPM이 빌드되는 기본 컴파일러 플래그에는 최적화 및 하드닝 플래그가 포함되어 있습니다. GCC로 자체 코드를 빌드하려면 최적화 및 하드닝 플래그를 포함하는 것이 좋습니다.

Note

gcc --version을 호출하면 gcc (GCC) 11.3.1 20221121 (Red Hat 11.3.1-4) 같은 버전 문자열이 표시됩니다. Red Hat은 Amazon Linux GCC 패키지의 기반이 되는 [GCC 공급업체 지점](#)을 나타냅니다. gcc --help에 표시된 버그 보고서 URL에 따라 모든 버그 신고 및 지원 요청은 Amazon Linux로 전달해야 합니다.

__GNUC_RH_RELEASE__ 매크로와 같은 이 공급업체 브랜치의 일부 장기 변경 사항에 대한 자세한 내용은 [Fedora 패키지 소스](#) 섹션을 참조하세요.

핵심 도구 체인에 대한 자세한 내용은 [코어 툴체인 패키지 glibc, gcc 및 binutils](#) 섹션을 참조하세요.

AL2023과 다른 Linux 배포판과의 관계에 대한 자세한 내용은 [Fedora와의 관계](#) 섹션을 참조하세요.

AL2와 비교한 AL2023 컴파일러 트리플렛 변경 사항에 대한 자세한 내용은 [컴파일러 트리플렛](#) 섹션을 참조하세요.

주제

- [GCC 14](#)
- [언어 표준 버전 비교](#)

GCC 14

AL2023은 기본 GCC 11과 함께 설치할 수 있는 선택적 컴파일러로 GCC 14를 제공합니다. GCC 14에는 최신 언어 기능 및 최적화가 포함되어 있으므로 최신 C, C++ 또는 Fortran 표준 지원이 필요한 프로젝트에 적합합니다.

GCC 14를 설치하려면 다음 명령을 사용하세요.

```
sudo dnf install gcc14 gcc14-c++ gcc14-gfortran
```

GCC 14 컴파일러는 기본 GCC 11과 충돌하지 않도록 버전별 명령 이름과 함께 설치됩니다.

- gcc14-gcc - C 컴파일러
- gcc14-g++ - C++ 컴파일러
- gcc14-gfortran - Fortran 컴파일러

사용 예:

```
gcc14-gcc -o myprogram myprogram.c
gcc14-g++ -o mycppprogram mycppprogram.cpp
gcc14-gfortran -o myfortranprogram myfortranprogram.f90
```

다음을 실행하여 설치한 버전을 검증합니다.

```
gcc14-gcc --version
```

그러면 gcc14-gcc (GCC) 14.2.1 20250110 (Red Hat 14.2.1-7)과 유사한 버전 정보가 표시됩니다.

Note

GCC 11과 GCC 14를 모두 동일한 시스템에 동시에 설치할 수 있습니다. 기본 gcc, g++ 및 gfortran 명령은 GCC 11을 계속 사용하는 반면, GCC 14에는 버전별 명령을 통해 액세스해야 합니다.

언어 표준 버전 비교

다음 표에서는 다양한 Amazon Linux 버전과 GCC 컴파일러 버전을 대상으로 기본 언어 표준 버전을 비교합니다.

Amazon Linux 버전	C 표준(기본값)	C++ 표준(기본값)	Fortran 표준
GCC 7을 포함한 AL2(기본값)	C11(201112L)	C++14(201402L)	Fortran 2008
GCC 10을 포함한 AL2(선택 사항)	C17/C18(201710L)	C++14(201402L)	Fortran 2008

Amazon Linux 버전	C 표준(기본값)	C++ 표준(기본값)	Fortran 표준
GCC 11을 포함한 AL2023(기본)	C17/C18(201710L)	C++17(201703L)	Fortran 2008
GCC 14를 포함한 AL2023(선택 사항)	C17/C18(201710L)	C++17(201703L)	Fortran 2008

GCC 버전별 주요 개선 사항:

- GCC 10 및 GCC 7: 기본 C 표준을 C11에서 C17/C18로 업그레이드하고, C++20 기능에 대한 지원을 추가하고, 최적화 기능을 개선했습니다.
- GCC 11 및 GCC 10: 기본 C++ 표준을 C++14에서 C++17로 업그레이드하고, C++20 지원을 강화하고, 실험적 C++23 기능을 추가했습니다.
- GCC 14 및 GCC 11: 전체 C23 표준 지원, 향상된 C++23 기능, 향상된 최적화, 향상된 표준 규정 준수가 추가되었습니다.

지원되는 언어 표준:

- C 표준: 모든 버전은 C90, C99, C11 및 C17/C18을 지원합니다. GCC 10+는 C2x(C23 초안)를 지원하는 반면, GCC 14는 전체 C23 지원을 제공합니다.
- C++ 표준: 모든 버전은 C++98, C++03, C++11, C++14, C++17 및 C++20을 지원합니다. GCC 11+는 실험적 C++23 지원을 제공하며, GCC 14는 향상된 C++23 기능을 제공합니다.
- Fortran 표준: 모든 버전은 주로 Fortran 2008을 지원하며, GCC 버전에 따라 다양한 수준의 Fortran 2018 기능을 제공합니다.

Note

기본 표준은 GCC 11~14 사이에서 일관되게 유지되지만, GCC 14는 `-std=` 플래그를 사용하여 명시적으로 요청할 때 언어 기능 지원, 최적화 개선, 진단 개선 및 보다 완전한 최신 표준 구현을 제공합니다.

AL2023에서 .NET 사용

AL2023은 현재 [.NET](#) 런타임 및 SDK 버전 8.0, 9.0 및 10.0을 제공합니다. .NET 6.0은 2024년 11월에 end-of-life에 도달했으며 더 이상 지원되지 않습니다. 각 버전은 업스트림과 동일한 기간 동안 지원됩니다. 자세한 내용은 [패키지 지원 설명서](#) 섹션을 참조하세요.

최신 .NET 버전으로 마이그레이션

업스트림 .NET 커뮤니티는 이동을 위한 마이그레이션 설명서를 제공합니다.

- [~ .NET 10](#)
- [~ .NET 9](#)
- [~ .NET 8](#)

AL2023에 .NET 설치

AL2023에는 여러 버전의 .NET 런타임을 동시에 설치할 수 있습니다. 그러나 dotnet 및 dotnet-host 패키지는 SDK 버전 간에 공유되므로 한 번에 하나의 SDK 버전만 /usr/bin/dotnet CLI 진입점을 소유할 수 있습니다. 다른 SDK 버전을 설치하면 공유 dotnet 명령이 해당 버전으로 업데이트됩니다.

SDK의 특정 버전을 설치하려면 dnf 명령을 .NET 사용합니다.

```
sudo dnf install dotnet-sdk-10.0
```

를 원하는 버전(8.0, 9.0 또는 10.0)10.0으로 바꿉니다.

SDK 없이 런타임만 설치하려면:

```
sudo dnf install dotnet-runtime-9.0
```

AL2023에서 Go 사용

Amazon Linux에서 [Go](#)로 작성된 자체 코드를 빌드하거나 AL2023의 도구 체인을 사용하여 빌드할 수 있습니다. AL2와 유사하게, AL2023은 운영 체제 수명 주기 동안 Go 도구 체인을 업데이트합니다. 당사는 제공하는 툴체인이 CVE에 대한 결과이거나 분기 릴리스에 포함되어 있습니다.

Go는 비교적 빠르게 움직이는 언어입니다. Go에 작성된 기존 애플리케이션이 새 버전의 Go 도구 체인에 적응해야 하는 상황이 있을 수 있습니다. Go에 대한 자세한 내용은 [Go 1 및 Go 프로그램의 미래](#) 섹션을 참조하세요.

AL2023은 수명 동안 새로운 버전의 Go 도구 체인을 통합하지만, 업스트림 Go 릴리스와 보조를 맞추지는 않습니다. 따라서 Go 언어 및 표준 라이브러리의 최첨단 기능을 사용하여 Go 코드를 빌드하는 경우 AL2023 Go 도구 체인을 사용하는 것은 적합하지 않을 수 있습니다.

AL2023 수명 동안 구 패키지 버전은 리포지토리에서 제거되지 않습니다. Go 도구 체인 구 버전이 필요한 경우 최신 Go 도구 체인의 버그 및 보안 수정을 생략하고 모든 RPM에서 사용할 수 있는 것과 동일한 메커니즘을 사용하여 리포지토리에서 구 버전을 설치할 수 있습니다.

AL2023에서 자체 Go 코드를 빌드하고 싶은 경우 AL2023에 포함된 해당 도구 체인을 사용할 수 있습니다. AL2023 수명 동안 Go 도구 체인은 계속 업그레이드됩니다.

Go에 작성된 AL2023 Lambda 함수

Go는 네이티브 코드로 컴파일되므로 Lambda는 Go를 사용자 지정 런타임으로 취급합니다. `provided.al2023` 런타임을 사용하여 AL2023의 Go 함수를 Lambda에 배포할 수 있습니다.

자세한 내용은 AWS Lambda 개발자 안내서의 [Go로 Lambda 함수 빌드](#) 섹션을 참조하세요.

AL2023에 설치된 Java

AL2023은 Java 기반 워크로드를 지원하는 다양한 버전의 [Amazon Corretto](#)를 제공합니다. AL2023에 포함된 모든 Java 기반 패키지는 모두 Amazon Corretto 17로 빌드됩니다.

Corretto는 Amazon의 장기 지원을 받는 공개 Java 개발 키트(OpenJDK)입니다. Corretto는 Java 기술 호환성 키트(TCK)로 Java SE 표준 인증을 받았으며 Linux, Windows 및 macOS에서 사용할 수 있습니다.

Corretto 1.8.0, Corretto 11, Corretto 17에 사용할 수 있는 [Amazon Corretto](#) 패키지가 있습니다.

AL2023에 설치된 Corretto는 Corretto 버전과 동일한 기간 동안 또는 AL2023 기간이 종료되는 시점 중 더 짧은 기간 동안 지원합니다. 자세한 내용은 [Amazon Linux 패키지 지원 설명서](#)와 [Amazon Corretto FAQ](#)를 참조하세요.

AL2023에서 Node.js 사용

[Node.js](#) AL2023의는 버전 20, 22 및 24로 표시됩니다. Amazon Linux는 업스트림 [지원 일정](#)을 따르며 모든 Node.js 버전의 지원 상태는 패키지 지원 [상태 페이지](#)에서 항상 확인할 수 있습니다. 지원되

는 모든 Node.js 버전은 네임스페이스가 지정되며 동일한 시스템에 동시에 설치할 수 있습니다. 네임스페이스 지정 시에는 각 Node.js 설치가 파일 시스템 내에서 고유하도록 합니다. 이는 런타임 버전을 기반으로 키 디렉터리와 파일의 이름을 변경하여 달성할 수 있습니다. 실제 실행 파일 이름은 node-{MAJOR_VERSION} 또는 npm-{MAJOR_VERSION}과 같습니다. 한 번에 Node.js 버전 하나만 활성화할 수 있습니다. 이 활성 버전은 node, npm 또는 /usr/lib/node_modules와 같은 기본 디렉터리와 파일 이름을 제공하며, 이는 현재 활성 런타임을 가리킵니다.

이는 대체 도구의 기능을 사용하여 이루어집니다. 기본 실행 파일 이름은 가상이며 설치된 다른 Node.js 버전을 가리킬 때 언제든지 변경될 수 있다는 점을 기억해야 합니다. 이러한 유연성을 통해 shebang의 노드를 사용하는 소프트웨어는 간접 호출 시 원하는 버전을 선택할 수 있습니다. 그러나 특정 버전의 Node.js가 필요한 경우 항상 지정된 버전의 런타임을 사용하는 네임스페이스 실행 파일(예: node-20 또는 node-22)을 호출하여 버전의 지속성을 달성할 수 있습니다. 또한 npm-20 또는 npm-22와 같은 npm 도구의 네임스페이스 실행 파일은 현재 활성 런타임에 관계없이 항상 해당 Node.js 버전과 연결됩니다.

Node.js는 "nodejs{MAJOR_VERSION}"로 시작하는 여러 네임스페이스 패키지로 배포됩니다. 이러한 패키지는 npm 도구, 설명서, 라이브러리 등의 호환 버전인 노드를 제공합니다. 예를 들어 22의 노드와 Node.js npm은 각각 nodejs22 및 nodejs22-npm 패키지에서 제공됩니다.

대체 도구는 Node.js 버전 간 전환을 위한 단일 명령을 제공합니다. 기본적으로 대체는 우선순위를 사용하여 현재 활성 Node.js 버전을 결정하는 자동 모드로 구성됩니다. 그러나 설치된 버전은 언제든지 활성화할 수 있습니다. 현재 Node.js의 지원되는 모든 버전은 우선순위가 동일합니다. 즉, 처음 설치된 버전이 자동으로 활성화됩니다.

대체를 사용하는 몇 가지 유용한 예제

1. 다음에 대해 구성된 대체 확인

```
alternatives --list
```

2. 노드의 현재 구성 확인

```
alternatives --display node
```

3. 대화형으로 Node.js 버전 변경

```
alternatives --config node
```

4. 수동 모드로 전환하고 특정 버전 선택

```
alternatives --set node /usr/bin/node-{MAJOR_VERSION}
```

5. 자동 버전 선택 모드로 다시 전환

```
alternatives --auto node
```

AL2023에 설치된 Perl

AL2023은 [Perl](#) 프로그래밍 언어 버전 5.32를 제공합니다.

Perl은 지난 수십 년 동안 Perl 5 릴리스의 언어로서 높은 수준의 호환성을 제공했지만, AL2023 릴리스 되는 동안 Amazon Linux가 Perl 5.32에서 이전할 것으로 예상하지는 않습니다. Amazon Linux는 [패키지 지원 설명서](#)에 따라 AL2023 서비스가 종료될 때까지 계속해서 Perl에 보안 패치를 적용합니다.

AL2023에 설치된 Perl 모듈

다양한 Perl 모듈이 AL2023에 RPM으로 패키징되어 있습니다. RPM으로 사용할 수 있는 Perl 모듈은 많지만 Amazon Linux는 가능한 모든 Perl 모듈을 패키징하는 것을 목표로 하지는 않습니다. RPM으로 패키징된 모듈은 다른 운영 체제 RPM 패키지에서 사용할 수 있으므로 Amazon Linux는 순수한 기능 업데이트보다 보안 패치를 우선시합니다.

AL2023에는 CPAN이 있어서 Perl 개발자가 Perl 모듈용 패키지 관리자를 사용할 수 있습니다.

AL2023에서 PHP 사용

AL2023은 현재 [PHP](#) 프로그래밍 언어 버전 8.1, 8.2, 8.3, 8.4 및 8.5를 제공합니다. 각 버전은 업스트림 PHP와 동일한 기간 동안 지원됩니다. 자세한 내용은 [패키지 지원 설명서](#) 섹션을 참조하세요.

PHP 구 버전에서 마이그레이션

업스트림 PHP 커뮤니티에서 다음과 같이 이전하기 위한 마이그레이션 종합 설명서를 만들었습니다.

- [PHP 8.4.x에서 PHP 8.5.x로](#)
- [PHP 8.3.x에서 PHP 8.4.x로](#)
- [PHP 8.2.x에서 PHP 8.3.x로](#)
- [PHP 8.1.x에서 PHP 8.2.x로](#)

- [PHP 8.0.x에서 PHP 8.1.x로](#)

AL2에는 amazon-linux-extras의 PHP 8.0, 8.1 및 8.2가 포함되어 AL2023 버전으로 쉽게 업그레이드할 수 있습니다.

PHP 7.x 버전에서 마이그레이션

Note

[PHP](#) 프로젝트에서 [지원하는 버전](#) 목록과 일정을 비롯해 [지원하지 않는 지점](#) 목록을 관리할 수 있습니다.

AL2023이 릴리스되었을 당시 PHP 커뮤니티는 [PHP](#) 7.x 및 5.x 버전을 지원하지 않았으며 AL2023 옵션으로도 포함되지 않았습니다.

업스트림 PHP 커뮤니티에서 [PHP 8.0에서 PHP 7.4로 이전하기 위한 마이그레이션 종합 설명서](#)를 만들었습니다. PHP 8.1 및 PHP 8.2로의 마이그레이션에 대한 이전 섹션에서 참조한 설명서와 함께 사용하면 PHP 기반 애플리케이션을 최신 PHP로 마이그레이션할 수 있습니다.

Note

AL2의 amazon-linux-extras에는 PHP 7.1, 7.2, 7.3 및 7.4가 포함되어 있습니다. 단, 이러한 엑스트라는 모두 서비스 종료된 제품이므로 추가 보안 업데이트를 보장하지 않습니다.

AL2023 PHP 모듈

AL2023에는 PHP Core에 포함된 PHP 모듈이 다수 있습니다. AL2023은 [PHP 확장 커뮤니티 라이브러리\(PECL\)](#)에 있는 패키지를 모두 사용하는 것을 목표로 하지 않습니다.

AL2023에서 Python 사용

AL2023에서 Python 2.7이 제거되었으며 Python 필요한 구성 요소는 이제 Python 3과 함께 작동합니다.

AL2023 버전에서는 Python 3를 /usr/bin/python3으로 사용할 수 있으며, 고객 코드 및 AL2023 버전과 함께 제공되는 Python 코드와 호환성을 유지하기 위해 AL2023이 지원 종료되기 전까지 Python 3.9로 유지됩니다.

/usr/bin/python3이 가리키는 python 버전은 시스템 Python'으로 간주되며, AL2023 버전의 경우 Python 3.9입니다.

AL2023은 현재 버전 3.11Python, 3.12, 3.13 및 3.14를 포함한 최신 버전의를 제공합니다. 각 버전은 업스트림과 동일한 기간 동안 지원됩니다Python. 자세한 내용은 [패키지 지원 설명서](#) 섹션을 참조하세요.

AL2023에 Python 여러 버전을 동시에 설치할 수 있습니다. /usr/bin/python3은 항상 Python 3.9이지만, Python 각 버전은 네임스페이스가 지정되어 있어서 버전 숫자로 찾을 수 있습니다. 예를 들어 python3.11이 설치되어 있으면 /usr/bin/python3.11은 /usr/bin/python3.9과 함께 존재하며 /usr/bin/python3은 /usr/bin/python3.9의 심볼릭 링크로 존재합니다.

Note

/usr/bin/python3 symlink가 가리키는 내용을 변경하지 마세요. AL2023의 핵심 기능을 손상시킬 수 있습니다.

최신 Python 버전으로 마이그레이션

업스트림 Python 커뮤니티는 이동을 위한 마이그레이션 설명서를 제공합니다.

- [Python ~ 3.14](#)
- [Python ~ 3.13](#)
- [Python ~ 3.12](#)
- [Python ~ 3.11](#)

AL2023 Python 모듈

다양한 Python 모듈이 AL2023에 RPM으로 패키징되어 있습니다. 일반적으로 Python 모듈의 RPM은 Python 시스템 버전만을 대상으로 빌드되었습니다.

AL2023에서 Ruby 사용

[Ruby](#) AL2023의는 버전 3.2 및 3.4로 표시됩니다. Amazon Linux는 업스트림 지원 일정을 따르며 [Ruby 웹 사이트에서](#) 항상 모든 Ruby 버전의 지원 상태를 확인할 수 있습니다. 지원되는 모든 Ruby 버전은 네임스페이스가 지정되며 동일한 시스템에 동시에 설치할 수 있습니다. 네임스페이스 지정 시에는 각 Ruby 설치가 파일 시스템 내에서 고유하도록 합니다. 이는 런타임 버전을 기반으로 키 디렉터리와 파

일의 이름을 변경하여 달성할 수 있습니다. 실제 실행 파일 이름은 ruby{MAJOR.MINOR}(예: ruby3.2 또는 ruby3.4)와 같습니다. Ruby 또한 3.4는의 표준 C 기반 참조 구현을 참조ruby3.4-mri하는 MRI(Matz의 Ruby 해석기) 네임스페이스 바이너리를 제공합니다Ruby. 한 번에 Ruby 버전 하나만 활성화할 수 있습니다. 이 활성 버전은 ruby, gem 또는 번들과 같은 기본 디렉터리와 파일 이름을 제공하여 현재 활성 런타임을 가리킵니다.

이는 대체 도구의 기능을 사용하여 이루어집니다. 기본 실행 파일 이름은 가상이며 설치된 다른 Ruby 버전을 가리킬 때 언제든지 변경될 수 있다는 점을 기억해야 합니다. 이러한 유연성을 통해 Shebang 에서 ruby를 사용하는 소프트웨어는 호출 시 원하는 버전을 선택할 수 있습니다. 그러나 특정 버전의 Ruby가 필요한 경우 항상 지정된 버전의 런타임을 사용하는 네임스페이스 실행 파일(예: ruby3.2 또는 ruby3.4)을 호출하여 버전의 지속성을 달성할 수 있습니다. 또한 , , ruby3.2-gem ruby3.4-gem ruby3.2-bundler또는와 같은 썬 및 번들러 도구의 네임스페이스 실행 ruby3.4-bundler파일은 현재 활성 런타임에 관계없이 항상 해당 Ruby 버전과 연결됩니다.

Ruby는 "ruby{MAJOR.MINOR}"로 시작하는 여러 네임스페이스 패키지로 배포됩니다. 이러한 패키지는 썬 및 번들러 도구, 설명서, 라이브러리 등의 ruby 호환 버전을 제공합니다. 예를 들어 코어 Ruby 3.2 런타임은 (썬 ruby3.2-rubygems 제공) 및 ruby3.2-rubygem-bundler (번들 및 번들러 제공)를 종속성으로 가져오는 ruby3.2 패키지에서 제공됩니다.

Ruby 버전을 설치한 후 대체 구성에서 컴패니언 도구의 항목이 null로 표시될 수 있습니다. 이를 실행하여 확인할 수 있습니다alternatives --display ruby. 항목이 null로 표시되면를 사용하여 수동으로 등록해야 합니다alternatives --install. 예를 들어 Ruby3.4에 대한 모든 컴패니언 도구를 등록하려면

```
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
```

우선 순위 값(예: 3.4의 경우 Ruby 34, 3.2의 경우 Ruby 32)은 기본 ruby 대체 항목에 사용된 우선 순위와 일치해야 합니다. 등록되면 ruby 대안과 함께 컴패니언 도구가 자동으로 관리됩니다.

대체 도구는 Ruby 버전 간 전환을 위한 단일 명령을 제공합니다. 기본적으로 대체는 우선순위를 사용하여 현재 활성 Ruby 버전을 결정하는 자동 모드로 구성됩니다. 그러나 설치된 버전은 언제든지 활성화할 수 있습니다. 현재 Ruby의 지원되는 모든 버전은 우선순위가 동일합니다. 즉, 처음 설치된 버전이 자동으로 활성화됩니다.

대체를 사용하는 몇 가지 유용한 예제

1. 다음에 대해 구성된 대체 확인

```
alternatives --list
```

2. ruby의 현재 구성 확인

```
alternatives --display ruby
```

3. 대화형으로 Ruby 버전 변경

```
alternatives --config ruby
```

4. 수동 모드로 전환하고 특정 버전 선택

```
alternatives --set ruby /usr/bin/ruby{MAJOR.MINOR}
```

5. 자동 버전 선택 모드로 다시 전환

```
alternatives --auto ruby
```

AL2023에서 Rust 사용

Amazon Linux에서 [Rust](#)로 작성된 자체 코드를 빌드하거나 AL2023의 도구 체인을 사용하여 빌드할 수 있습니다.

AL2와 유사하게, AL2023은 운영 체제 수명 주기 동안 Rust 도구 체인을 업데이트합니다. 이는 당사가 제공하는 툴체인이 CVE에 대한 결과이거나 분기 릴리스에 포함되어 있습니다.

[Rust](#)는 비교적 빠르게 업데이트되는 언어로서, 약 6주마다 새 릴리스가 출시됩니다. 이러한 릴리스에 새로운 언어 또는 표준 라이브러리 기능이 추가될 수 있습니다. AL2023은 수명 동안 새로운 버전의 Rust 도구 체인을 통합하지만, 업스트림 Rust 릴리스와 보조를 맞추지는 않습니다. 따라서 Rust 언어의 최첨단 기능을 사용하여 Rust 코드를 빌드하는 경우 AL2023 Rust 도구 체인을 사용하는 것은 적합하지 않을 수 있습니다.

AL2023 서비스 기간 동안 패키지 구 버전은 리포지토리에서 제거되지 않습니다. Rust 도구 체인 구 버전이 필요한 경우 최신 Rust 도구 체인의 버그 및 보안 수정을 생략하고 모든 RPM에서 사용할 수 있는 것과 동일한 메커니즘을 사용하여 리포지토리에서 구 버전을 설치할 수 있습니다.

AL2023에서 자체 Rust 코드를 빌드하고 싶은 경우 AL2023에 포함된 해당 도구 체인을 사용할 수 있습니다. AL2023 수명 동안 Rust 도구 체인은 계속 업그레이드됩니다.

Rust에 작성된 AL2023 Lambda 함수

Rust는 네이티브 코드로 컴파일되므로 Lambda는 Rust를 사용자 지정 런타임으로 취급합니다. `provided.al2023` 런타임을 사용하여 AL2023의 Rust 함수를 Lambda에 배포할 수 있습니다.

자세한 내용은 AWS Lambda 개발자 안내서의 [Rust로 Lambda 함수 빌드](#) 섹션을 참조하세요.

AL2023에서 Swift 사용

AL2023은 현재 [Swift](#) 런타임 및 SDK를 제공합니다.

AL2023에서는 Swift SDK와 런타임이 별도의 패키지로 분할됩니다. `swiftlang-lib` 패키지에는 런타임 라이브러리가 포함되어 있고 `swiftlang` 패키지에는 전체 SDK가 포함되어 있습니다. 이를 통해 사용자는 전체 Swift SDK를 설치할 필요 없이 프로덕션 호스트에서 SDK로 컴파일된 바이너리를 실행할 수 있습니다. 현재의 다른 버전 Swift은 ABI와 호환되지 않으므로 업데이트할 때 Swift애플리케이션을 배포하려면 먼저 새 버전으로 애플리케이션을 다시 빌드해야 합니다.

Note

와 마찬가지로 Rust Swift 버전은 Swift 프로젝트의 안정적인 최신 버전을 따릅니다. 패키지는 네임스페이스가 지정되지 않으며 한 번에 하나의 버전만 지원됩니다.

AL2023Swift에서 사용에 대한 몇 가지 참고 사항

- C++ 상호 운용성 - 시스템의 패키지 관리자가에서 작동하지 않을 `clang` 수 있습니다Swift. 이 문제를 해결하려면 `swift-clang`에를 추가PATH하거나 대체를 사용하여를 `clang`로 설정합니다`swift-clang`.
- Swift 코드 디버깅 - 시스템에서 Swift 소스를 디버깅LLDB할 수 없습니다. 대신에 Swift 설치 디렉토리를 추가PATH하거나 대체를 사용하여 Swift 패키지에 LLDB 포함된를 사용하여를 `lldb`로 설정합니다`swift-lldb`.

TypeScriptAL2023에서 사용

Note

이 문서는 TypeScript 및 Node.js 기반 실행 환경에 대한 필수 정보를 제공합니다. 또한 일반적인 개발 워크플로를 다루고 TypeScript가 AL2023에 패키징되어 일관되고 재현 가능한 개발 환경을 제공하는 방법을 설명합니다.

[TypeScript](#)(TS)는 모든 JS의 기능을 제공하고 [유형 시스템으로 확장](#)하는 JavaScript(JS) 기반 프로그래밍 언어입니다. 일반적인 시나리오에서 TS로 작성된 프로그램은 먼저 JS 코드로 변환된 다음 Node.js에 의해 다른 일반 JS 프로그램으로 실행됩니다. TS의 특정 용어에서 이 번역 프로세스를 '[컴파일](#)'이라고 하며 tsc라는 '컴파일러'에 의해 수행됩니다. tsc 컴파일러 자체는 JS로 작성되므로 실행하려면 Node.js 같은 JS 런타임 환경도 필요합니다. 일부 다른 JS 런타임 환경과 달리 Node.js는 현재 실험적인 경량 TS만 지원합니다. 유형 확인을 포함한 전체 TS 지원에는 여전히 [typescript](#) 같은 타사 패키지를 사용해야 합니다. Node.js 런타임 환경에 대한 tsc(TS 컴파일러)를 가져오는 예상 방법은 typescript 노드 모듈을 설치하는 것입니다. 이는 일반적으로 npm이라는 패키지 관리자 중 하나를 사용하여 수행할 수 있습니다. npm을 사용하여 TS 컴파일러를 설치하는 방법에는 글로벌 및 프로젝트 등 두 가지가 있습니다. [공식적으로 권장되는 방법은 프로젝트별로 TS 컴파일러를 설치](#)하여 프로젝트의 장기 일관성과 재현성을 보장하는 것입니다. 그러나 TS 컴파일러를 글로벌로 설치하는 것은 전체 호스트 및 JS 런타임에 대해 동일한 버전을 제공하므로 로컬에 TS 컴파일러가 설치되지 않은 프로젝트에도 여전히 유용할 수 있습니다. 이는 nodejs20-typescript 또는 nodejs22-typescript 같이 Amazon Linux에서 사용할 수 있는 RPM 패키지가 시스템 수준에서 글로벌로 지원되는 각 Node.js 버전에 대해 별도로 TS 컴파일러를 설치하는 정확한 방법입니다.

tsc는 어떤 Node.js 버전에도 직접 종속되지 않습니다. 컴파일러는 [target](#) 및 [lib](#)와 같은 옵션을 통해 특수 파일(tsconfig.json)에 정의된 특정 수준의 런타임 기능을 기대합니다. 이러한 옵션의 값은 JS 런타임 환경에서 지원할 수 있는(또는 지원하지 않을 수 있는) [ECMAScript](#)(ES) 표준의 버전을 나타냅니다. Node.js 버전에 따라 지원되는 ES 표준 버전이 다릅니다. Node.js 버전이 최신일수록 지원되는 ES 표준 버전이 더 높고 더 나은 완전성을 가집니다. tsconfig.json이 프로젝트의 루트 디렉터리에 없는 경우 기본 구성 옵션 세트가 사용됩니다. Node.js의 다양한 버전에 대한 호환성 테이블과 다양한 ES 표준 버전의 지원되는 기능은 [node.green](#) 섹션에서 확인할 수 있습니다. tsc에는 tsconfig.json에 정의할 수 있는 100개 이상의 다양한 옵션이 있습니다. 일부 구성 옵션이 다른 파일에 정의된 다음 기본 파일에 포함된 경우에도 구성 체인이 지원됩니다. 이 접근 방식을 사용하면 특정 버전의 Node.js와 호환되는 [기본 TS 구성](#)을 설치한 다음 프로젝트별 옵션으로 확장할 수 있습니다. 다행히 Node.js용 기본 TS 구성은 npm을 사용하여 프로젝트 폴더에 설치할 수 있는 노드 모듈로 사용할 수 있습니다. 다음은 Node.js 버전 [18](#), [20](#) 및 [22](#)에 대한 구성의 소스 코드입니다.

Node.js 기반 런타임 설계에는 특정 약점이 있습니다. 즉, 호스트에서 하나의 런타임 버전만 지원하며 프로젝트 수준에서 모든 종속성의 재현성과 일관성이 필요합니다. 이로 인해 TypeScript를 사용하는 다음과 같은 일반적인 접근 방식이 생겼습니다. TS 컴파일러, 현재 Node.js 버전의 TS 기본 구성, 모든 소프트웨어 종속성이 프로젝트 내에 로컬로 설치됩니다. 글로벌로 설치된 노드 모듈은 npm 같은 CLI 도구일 것으로 예상되지만, CLI 도구이기도 한 tsc가 글로벌로 설치되는 경우는 거의 없습니다. 다행히 tsc의 글로벌(시스템 전체) 및 로컬(프로젝트 내) 설치의 문제 없이 공존할 수 있으며 독립적으로 사용되는 다른 버전일 수도 있습니다. 로컬로 설치된 tsc는 npm과 함께 설치되는 npx 도구를 사용하여 실행해야 합니다. 따라서 시스템 TS 컴파일러를 사용하더라도 사용자는 Node.js(대체를 통해 활성 버전을 전환), TS 컴파일러(로컬 또는 글로벌로 설치 후 대체를 통해 활성 버전 전환) 등 런타임 구성 요소의 버전하고 특정 요구 사항에 맞게 구성할 수 있습니다.

Amazon Linux는 npm 같이 글로벌로 설치된 다른 노드 모듈과 동일한 방식으로 Node.js 버전별로 TS 컴파일러를 패키징합니다. 패키지 및 바이너리는 네임스페이스가 지정되며 이름의 일부로 Node.js의 메이저 버전을 포함합니다. 컴파일러의 기본 실행 파일 이름인 tsc는 대체 도구에 의해 런타임 시 관리되며 설치되어 실행될 Node.js의 현재 활성 버전을 가리킵니다. 이 선택은 현재 Node.js 런타임 버전에 종속되지 않습니다. 노드 실행 파일이 Node.js 20을 가리키고 tsc가 Node.js 22로 해석되도록 구성할 수 있습니다. TS 컴파일러의 네임스페이스 이름(예: tsc-{MAJOR_VERSION})을 기본 tsc 이름 구성과 독립적으로 사용할 수도 있습니다.

TS 컴파일러의 활성 버전을 관리하는 데 유용한 몇 가지 명령

1. 다음에 대해 구성된 대체 확인

```
alternatives --list
```

2. tsc의 현재 구성 확인

```
alternatives --display tsc
```

3. 대화형으로 tsc 버전 변경

```
alternatives --config tsc
```

4. 수동 모드로 전환하고 특정 버전 선택

```
alternatives --set tsc /usr/bin/tsc-{MAJOR_VERSION}
```

5. 자동 버전 선택 모드로 다시 전환

```
alternatives --auto tsc
```

여러 버전의 노드와 TS 컴파일러를 동일한 시스템에 설치하고 사용하는 예제:

```
# 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 예약 사용자 및 그룹

AL2023은 이미지 프로비저닝 및 특정 패키지 설치 중에 특정 사용자 및 그룹을 사전 할당합니다. 충돌을 방지하기 위해 사용자, 그룹 및 관련 UID와 GID가 여기에 나열됩니다.

주제

- [AL2023 예약 사용자 목록](#)
- [AL2023 예약 그룹 목록](#)

AL2023 예약 사용자 목록

사용자 이름	UID
root	0
bin	1
daemon	2
adm	3
lp	4
동기화	5
shutdown	6
중지	7
메일	8
연산자	11
게임	12
ftp	14
squid	23

사용자 이름	UID
named	25
postgres	26
mysql	27
nscd	28
nscd	28
rpcuser	29
rpc	32
mailnull	47
apache	48
smmsp	51
tomcat	53
ldap	55
tss	59
nslcd	65
avahi	70
tcpdump	72
sshd	74
radvd	75
dbus	81
PostFix	89

사용자 이름	UID
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
memcached	976

사용자 이름	UID
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
systemd-coredump	996

사용자 이름	UID
libstoragemgmt	997
systemd-oom	999
ec2-user	1000
nobody	65534

이름별로 나열됨

사용자 이름	UID
adm	3
apache	48
avahi	70
avahi-autoipd	170
aws-kinesis-agent-user	989
bin	1
chrony	994
clamilt	985
clamscan	984
clamupdate	986
colord	987
cwagent	991
daemon	2

사용자 이름	UID
dbus	81
debuginfod	980
dnsmasq	982
dovecot	97
dovnull	981
ec2-instance-connect	993
ec2-user	1000
flatpak	979
ftp	14
게임	12
중지	7
haproxy	978
ldap	55
libstoragemgmt	997
lp	4
메일	8
mailnull	47
memcached	976
munge	975
mysql	27

사용자 이름	UID
named	25
nginx	974
nobody	65534
nscd	28
nscd	28
nslcd	65
ods	988
연산자	11
pesign	971
polkitd	992
PostFix	89
postgres	26
pulse	171
radvd	75
redis6	970
root	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179

사용자 이름	UID
saslauth	990
shutdown	6
smmsp	51
smtpd	973
smtpq	972
sphinx	977
squid	23
sshd	74
stap-server	962
stapdev	158
stapsys	157
stapusr	156
동기화	5
systemd-coredump	996
systemd-journal-remote	963
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
tcpdump	72

사용자 이름	UID
tomcat	53
tss	59
unbound	983
uidd	961

AL2023 예약 그룹 목록

그룹 이름	GID
root	0
bin	1
daemon	2
sys	3
adm	4
tty	5
디스크	6
디스크	6
lp	7
mem	8
kmem	9
wheel	10
cdrom	11

그룹 이름	GID
메일	12
메일	12
man	15
dialout	18
floppy	19
게임	20
slocate	21
utmp	22
squid	23
named	25
postgres	26
mysql	27
nscd	28
nscd	28
rpcuser	29
rpc	32
테이프	33
utempter	35
kvm	36
비디오	39

그룹 이름	GID
mailnull	47
apache	48
ftp	50
smmsp	51
tomcat	53
잠금	54
ldap	55
tss	59
오디오	63
avahi	70
tcpdump	72
sshd	74
radvd	75
saslauth	76
dbus	81
screen	84
wbpriv	88
PostFix	89
postdrop	90
dovecot	97

그룹 이름	GID
사용자	100
입력	104
render	105
sgx	106
mock	135
stapusr	156
stapusr	156
stapsys	157
stapsys	157
stapdev	158
stapdev	158
avahi-autoipd	170
pulse	171
rtkit	172
sanlock	179
systemd-journal	190
systemd-network	192
systemd-resolve	193
usbmon	959
wireshark	960

그룹 이름	GID
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
추적	973
haproxy	974
flatpak	975
debuginfod	976
dovnull	977
dnsmasq	978
unbound	979
clamscan	980

그룹 이름	GID
clamilt	981
virusgroup	982
virusgroup	982
virusgroup	982
clamupdate	983
printadmin	984
colord	985
ods	986
Docker	987
aws-kinesis-agent-user	988
cwagent	989
pulse-rt	990
pulse-access	991
ec2-instance-connect	993
chrony	994
systemd-timesync	995
systemd-coredump	996
libstoragemgmt	997
ssh_keys	998
systemd-oom	999

그룹 이름	GID
ec2-user	1000
ne	1001
polkitd	9920
nobody	65534

이름별로 나열됨

그룹 이름	GID
adm	4
apache	48
오디오	63
avahi	70
avahi-autoipd	170
aws-kinesis-agent-user	988
bin	1
cdrom	11
chrony	994
clamilt	981
clamscan	980
clamupdate	983
colord	985

그룹 이름	GID
cwagent	989
daemon	2
dbus	81
debuginfod	976
dialout	18
디스크	6
디스크	6
dnsmasq	978
Docker	987
dovecot	97
dovnull	977
ec2-instance-connect	993
ec2-user	1000
flatpak	975
floppy	19
ftp	50
게임	20
haproxy	974
입력	104
kmem	9

그룹 이름	GID
kvm	36
ldap	55
libstoragemgmt	997
잠금	54
lp	7
메일	12
메일	12
mailnull	47
man	15
mem	8
memcached	971
mock	135
munge	970
mysql	27
named	25
ne	1001
nginx	969
nobody	65534
nscd	28
nscd	28

그룹 이름	GID
ods	986
pesign	966
polkitd	9920
postdrop	90
PostFix	89
postgres	26
printadmin	984
pulse	171
pulse-access	991
pulse-rt	990
radvd	75
redis6	965
render	105
root	0
rpc	32
rpcuser	29
rtkit	172
sanlock	179
saslauth	76
screen	84

그룹 이름	GID
sgx	106
slocate	21
smmsp	51
smtpd	968
smtpq	967
sphinx	972
squid	23
ssh_keys	998
sshd	74
stap-server	962
stapdev	158
stapdev	158
stapsys	157
stapsys	157
stapusr	156
stapusr	156
sys	3
systemd-coredump	996
systemd-journal	190
systemd-journal-remote	963

그룹 이름	GID
systemd-network	192
systemd-oom	999
systemd-resolve	193
systemd-timesync	995
테이프	33
tcpdump	72
tomcat	53
추적	973
tss	59
tty	5
unbound	979
usbmon	959
사용자	100
usershares	964
utempter	35
utmp	22
uidd	961
비디오	39
virusgroup	982
virusgroup	982

그룹 이름	GID
virusgroup	982
wbpriv	88
wheel	10
wireshark	960

AL2023에서 사용 가능한 코덱 목록

AL2023은 표준 리포지토리를 통해 다양한 멀티미디어 코덱을 제공합니다. 이 페이지에서는 코덱과 일반적인 사용 사례에 대한 개요를 제공합니다.

Important

Amazon Linux에 포함된 코덱을 사용하고 배포하려면 특정 타사 오디오 및 비디오 형식의 소유자 또는 라이선스 제공자를 포함한 타사로부터 라이선스 권한을 얻어야 할 수 있습니다. 이러한 라이선스를 획득하고 필요한 로열티 또는 요금을 지불할 책임은 전적으로 사용자에게 있습니다.

코덱	설명
flac	데이터나 품질을 잃지 않고 오디오를 압축하는 무료 오픈 소스 무손실 오디오 코덱으로, 일반적으로 고품질 오디오 스토리지에 사용됩니다.
fdk-aac-free	스트리밍 또는 파일 스토리지와 같은 MP3 대안에 고품질 오디오 압축을 제공하는 고급 오디오 코덱(AAC) 표준의 오픈 소스 구현
webrtc-audio-processing	웹 실시간 통신(WebRTC)에 사용되는 오디오 처리를 위한 라이브러리로, 노이즈 억제, 에코 취소 및 이득 제어와 같은 기능을 제공합니다.
opus	실시간 스트리밍용으로 설계된 매우 다재다능하고 효율적인 오디오 코덱으로, 짧은 지연 시간을 제공하고 VoIP 및 음악 스트리밍을 포함한 다양한 오디오 애플리케이션을 지원합니다.
libsndfile	오디오 처리 및 조작 도구에 일반적으로 사용되는 다양한 형식(예: WAV, AIFF 및 FLAC)의 오디오 파일을 읽고 쓰기 위한 라이브러리
openh264	H.264 비디오 코덱의 오픈 소스 구현입니다. 스트리밍, 회의 및 멀티미디어 애플리케이션에 사

코덱	설명
	용되는 H.264 비디오에 대한 인코딩 및 디코딩 지원을 제공합니다.
svt-av1	AV1 비디오 코덱의 오픈 소스 고성능 구현입니다. 최신 CPU 및 병렬 처리에 최적화된 AV1 비디오에 대한 확장 가능한 인코딩 및 디코딩을 제공합니다. CPUs
dav1d	속도, 효율성 및 이동성에 중점을 둔 AV1 비디오 디코더의 오픈 소스 구현입니다. 다양한 플랫폼 및 애플리케이션을 위한 고성능 AV1 디코딩을 제공합니다.
libde265	H.265/HEVC 비디오 디코더의 오픈 소스 구현입니다. 멀티미디어 애플리케이션 및 프레임워크에 사용할 수 있는 HEVC 비디오 스트림의 효율적이고 이식 가능한 디코딩을 제공합니다.
libheif	HEIF 및 AVIF 이미지 파일을 읽고 쓰기 위한 오픈 소스 라이브러리입니다. HEVC 및 AV1과 같은 최신 압축 형식을 사용하여 이미지 및 이미지 시퀀스를 효율적으로 인코딩, 디코딩 및 변환할 수 있습니다.
mpg123	MP3 파일을 포함한 MPEG 오디오 스트림용 고성능 디코더입니다. 재생, 스트리밍 및 오디오 처리 애플리케이션을 위한 빠르고 정확한 오디오 디코딩을 제공합니다.
x265	x265의 기본 목표는 어디서나 사용할 수 있는 최상의 H.265/HEVC 인코더가 되어 다양한 하드웨어 플랫폼에서 최고의 압축 효율성과 최고의 성능을 제공하는 것입니다. 이 패키지에는 명령줄 인코더가 포함되어 있습니다.

Amazon Linux 2023 보안 및 규정 준수

Important

취약성을 보고하거나 AWS 클라우드 서비스 또는 오픈 소스 프로젝트와 관련된 보안 문제가 있는 경우 [취약성 보고 페이지](#)를 사용하여 AWS 보안에 문의하세요.

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS와 사용자 간의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호할 책임이 있습니다. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 [AWS 규정 준수 프로그램](#) 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. AL2023에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 [AWS 제공 범위 내 서비스 규정 준수 프로그램](#).
- 클라우드 내 보안 - 귀하의 책임은 귀하가 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 데이터의 민감도, 회사 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

주제

- [AL2023용 Amazon Linux 보안 권고](#)
- [적용 가능한 권고 사항 나열](#)
- [현재 위치에서 보안 업데이트 적용](#)
- [AL2023의 SELinux 모드 설정](#)
- [AL2023에서 FIPS 모드 활성화](#)
- [AL2023 컨테이너에서 FIPS 모드 활성화](#)
- [AL2023에서 OpenSSL FIPS 공급자 전환](#)
- [AL2023 커널 하드닝](#)
- [AL2023에서 리포지토리 메타데이터 서명](#)
- [AL2023의 UEFI 보안 부팅](#)

AL2023용 Amazon Linux 보안 권고

Amazon Linux의 보안을 유지하기 위해 최선을 다하고 있지만 때로는 해결이 필요한 보안 문제가 있을 수 있습니다. 수정이 있을 때 권고 사항이 발행됩니다. Amazon Linux 보안 센터(ALAS)에 권고 사항이 게시되는 기본 위치입니다. 자세한 정보는 [Amazon Linux 보안 센터](#)를 참조하세요.

Important

취약성을 보고하거나 AWS 클라우드 서비스 또는 오픈 소스 프로젝트와 관련된 보안 문제가 있는 경우 [취약성 보고 페이지](#)를 사용하여 AWS 보안에 문의하세요.

Amazon Linux 팀은 AL2023에 영향을 미치는 문제와 관련 업데이트에 대한 정보를 여러 곳에 게시합니다. 보안 도구가 이러한 기본 소스에서 정보를 가져와 결과를 제공합니다. 따라서 Amazon Linux에서 게시하는 기본 소스와 직접 상호 작용하는 것이 아니라 [Amazon Inspector](#) 같이 선호하는 도구가 제공하는 인터페이스와 상호 작용합니다.

Amazon Linux 보안 센터 공지

Amazon Linux 공지는 권고에 맞지 않는 항목에 대해 제공됩니다. 이 섹션에는 권고에 맞지 않는 정보와 함께 ALAS 자체에 대한 공지가 포함되어 있습니다. 자세한 정보는 [Amazon Linux 보안 센터\(ALAS\)](#) 섹션을 참조하세요.

예를 들어 [2021-001 - Apache Log4j용 Amazon Linux 핫패치 공지](#)는 권고가 아닌 공지에 적합합니다. 이번 공고에서 Amazon Linux는 고객이 Amazon Linux에 속하지 않은 소프트웨어의 보안 문제를 완화하는 데 도움이 되는 패키지를 추가했습니다.

[Amazon Linux 보안 센터 CVE 탐색기](#)도 ALAS 공지 사항에서 게시되었습니다. 자세한 내용은 [CVE에 관한 새 웹사이트](#) 섹션을 참조하세요.

Amazon Linux 보안 센터 자주 묻는 질문

[Amazon Linux 보안 센터\(ALAS\) 자주 묻는 질문\(FAQ\)](#)에는 ALAS와 Amazon Linux가 CVE를 평가하는 방법에 대한 자주 묻는 질문의 답변이 있습니다.

ALAS 권고

Amazon Linux 권고에는 Amazon Linux 사용자와 관련된 중요한 정보, 일반적으로 보안 업데이트에 대한 정보가 포함되어 있습니다. [Amazon Linux 보안 센터](#)에서는 웹에 권고 사항이 표시됩니다. 권고 정보도 RPM 패키지 리포지토리 메타데이터의 일부입니다.

권고 사항 및 RPM 리포지토리

Amazon Linux 2023 패키지 리포지토리에는 0개 이상의 업데이트를 설명하는 메타데이터가 포함될 수 있습니다. `dnf updateinfo` 명령은 이 정보가 포함된 리포지토리 메타데이터 파일 이름의 이름을 따서 `updateinfo.xml`로 명명됩니다. 명령의 이름은 `updateinfo`이고 메타데이터 파일은 `update`를 참조하지만, 모두 권고의 일부인 패키지 업데이트를 참조합니다.

Amazon Linux 권고는 `dnf` 패키지 관리자가 참조하는 RPM 리포지토리 메타데이터에 있는 정보와 함께 [Amazon Linux 보안 센터](#) 웹사이트에 게시됩니다. 웹사이트 및 리포지토리 메타데이터는 최종적으로 일관되며 웹사이트 및 리포지토리 메타데이터의 정보에 불일치가 있을 수 있습니다. 이는 일반적으로 AL2023의 새 릴리스가 릴리스되는 동안 최신 AL2023 릴리스 이후 권고가 업데이트되었을 때 발생합니다.

문제를 해결하는 패키지 업데이트와 함께 새 권고를 발행하는 것이 일반적이지만 항상 그렇지 않습니다. 이미 릴리스된 패키지에서 해결되는 새 문제에 대해 권고를 생성할 수 있습니다. 기존 업데이트에서 다루는 새 CVEs로 기존 권고를 업데이트할 수도 있습니다.

Amazon Linux 2023의 [AL2023에서 버전 관리 리포지토리를 통한 결정적 업그레이드](#) 기능은 특정 AL2023 버전의 RPM 리포지토리에 해당 버전의 RPM 리포지토리 메타데이터 스냅샷이 포함되어 있음을 의미합니다. 여기에는 보안 업데이트를 설명하는 메타데이터가 포함됩니다. 특정 AL2023 버전의 RPM 리포지토리는 릴리스 후 업데이트되지 않습니다. 이전 버전의 AL2023 RPM 리포지토리를 볼 때는 새 보안 권고 또는 업데이트된 보안 권고가 표시되지 않습니다. `dnf` 패키지 관리자를 사용하여 `latest` 리포지토리 버전 또는 특정 AL2023 릴리스를 확인하는 방법은 [적용 가능한 권고 사항 나열](#) 섹션을 참조하세요.

권고 ID

각 권고는 `id`에서 참조합니다. 현재는 [Amazon Linux 보안 센터](#) 웹사이트에서 권고를 [ALAS-2024-581](#)로 나열하고 `dnf` 패키지 관리자는 해당 권고를 [ALAS2023-2024-581의 ID를 가진 것으로 나열](#)하는 Amazon Linux 쿼크입니다. [현재 위치에서 보안 업데이트 적용](#)의 경우 특정 권고를 참조할 때 패키지 관리자 ID를 사용해야 합니다.

Amazon Linux의 경우 OS의 각 메이저 버전에는 자체 네임스페이스인 권고 ID가 있습니다. Amazon Linux 권고 ID의 형식에 대해 어떠한 가정도 해서는 안 됩니다. 과거에 Amazon Linux 권고 ID는 NAMESPACE-YEAR-NUMBER 패턴을 따랐습니다. NAMESPACE에 사용할 수 있는 값의 전체 범위는 정의되지 않았지만 ALAS, ALASCORRETT08, ALAS2023, ALAS2, ALASPYTHON3.8 및 ALASUNBOUND-1.17이 포함되어 있습니다. YEAR는 권고가 생성된 연도이며 NUMBER는 네임스페이스 내의 고유 정수입니다.

권고 ID는 일반적으로 순차적이며 업데이트가 릴리스되는 순서대로 제공되지만, 그럴 수 없는 이유는 많으므로 이를 가정해서는 안 됩니다.

권고 ID를 Amazon Linux의 각 메이저 버전에 고유한 불투명 문자열로 취급합니다.

Amazon Linux 2에서는 각 Extra가 별도의 RPM 리포지토리에 있었으며, 권고 메타데이터는 관련 리포지토리 내에만 포함되어 있습니다. 한 리포지토리에 대한 권고는 다른 리포지토리에 적용할 수 없습니다. [Amazon Linux 보안 센터](#) 웹사이트에는 현재 각 주요 Amazon Linux 버전에 대한 권고 목록이 하나 있으며, 이는 리포지토리별 목록으로 구분되지 않습니다.

AL2023은 Extras 메커니즘을 사용하여 대체 버전의 패키지를 패키징하지 않으므로 현재 RPM 리포지토리가 두 개뿐이며, 각 리포지토리에는 core 리포지토리와 livepatch 리포지토리가 있습니다. livepatch 리포지토리는 [AL2023의 커널 라이브 패치](#)에 대한 것입니다.

권고 릴리스 날짜 및 권고 업데이트 날짜

Amazon Linux 권고의 권고 릴리스 날짜는 RPM 리포지토리에서 보안 업데이트가 처음 공개된 시기를 나타냅니다. 권고 사항은 수정 사항을 RPM 리포지토리를 통해 설치할 수 있게 된 직후 [Amazon Linux 보안 센터](#) 웹사이트에 게시됩니다.

권고 업데이트 날짜는 새 정보가 이전에 게시된 후 권고에 추가된 시기를 나타냅니다.

AL2023 버전 번호(예: 2023.6.20241031)와 해당 릴리스와 함께 게시된 권고 릴리스일 사이에는 어떠한 가정도 해서는 안 됩니다.

권고 유형

RPM 리포지토리 메타데이터는 다양한 유형의 권고를 지원합니다. Amazon Linux에는 보안 업데이트인 거의 보편적으로만 발급된 권고가 있지만 이를 가정해서는 안 됩니다. 버그 수정, 개선 사항 및 새 패키지와 같은 이벤트에 대한 권고가 발행되고 권고가 해당 유형의 업데이트가 포함된 것으로 표시될 수 있습니다.

권고 심각도

각 문제는 별도로 평가되므로 각 권고에는 고유한 심각도가 있습니다. 여러 CVE가 단일 권고에서 처리될 수 있으며 각 CVE는 다른 평가를 가질 수 있지만 권고 자체에는 하나의 심각도가 있습니다. 단일 패키지 업데이트를 참조하는 여러 권고 사항이 있을 수 있으므로 특정 패키지 업데이트의 심각도가 상이할 수 있습니다(권고 사항당 하나).

심각도를 낮추기 위해 Amazon Linux는 심각, 중요, 중간 및 낮음을 사용하여 권고의 심각도를 표시했습니다. 매우 드물긴 하지만 Amazon Linux 권고에 심각도가 없을 수도 있습니다.

Amazon Linux는 보통(Moderate)이라는 용어를 사용하는 RPM 기반 Linux 배포판 중 하나이며, 일부 다른 RPM 기반 Linux 배포판은 동등한 용어로 보통(Medium)을 사용합니다. Amazon Linux 패키지 관리자는 두 용어를 동등한 것으로 취급하며, 타사 패키지 리포지토리는 보통(Medium)이라는 용어를 사용할 수 있습니다.

Amazon Linux 권고는 권고에서 해결된 관련 문제에 대해 더 많이 알게 되면 시간이 지남에 따라 심각도를 변경할 수 있습니다.

권고의 심각도는 일반적으로 권고에서 참조하는 CVE에 대해 가장 높은 Amazon Linux 평가 CVSS 점수를 추적합니다. 그렇지 않은 경우가 있을 수 있습니다. CVE가 할당되지 않은 문제가 해결되는 경우를 예로 들 수 있습니다.

Amazon Linux에서 권고 심각도 등급을 사용하는 방법에 대한 자세한 내용은 [ALAS FAQ](#) 섹션을 참조하세요.

권고 및 패키지

단일 패키지에 대해 많은 권고가 있을 수 있으며 모든 패키지에 권고가 게시되는 것은 아닙니다. 특정 패키지 버전은 여러 권고에서 참조될 수 있으며, 각 권고는 고유한 심각도 및 CVE를 가집니다.

동일한 패키지 업데이트에 대한 여러 권고가 하나의 새로운 AL2023 릴리스에서 동시에 또는 빠르게 연속적으로 발행될 수 있습니다.

다른 Linux 배포판과 마찬가지로 동일한 소스 패키지에서 빌드된 하나 이상의 다양한 바이너리 패키지가 있을 수 있습니다. 예를 들어 [ALAS-2024-698](#)은 [Amazon Linux 보안 센터 웹사이트의 AL2023 섹션](#)에 mariadb105 패키지에 적용되는 것으로 나열된 권고 사항입니다. 이는 소스 패키지 이름이며, 권고 자체는 소스 패키지와 함께 바이너리 패키지를 참조합니다. 이 경우 12개 이상의 바이너리 패키지가 하나의 mariadb105 소스 패키지에서 빌드됩니다. 소스 패키지와 이름이 같은 바이너리 패키지가 있는 것은 매우 일반적이지만 범용은 아닙니다.

Amazon Linux 권고는 일반적으로 업데이트된 소스 패키지에서 빌드된 모든 바이너리 패키지를 나열하지만 이를 가정해서는 안 됩니다. 패키지 관리자 및 RPM 리포지토리 메타데이터 형식을 사용하면 업데이트된 바이너리 패키지의 하위 집합을 나열하는 권고를 사용할 수 있습니다.

특정 권고 사항은 특정 CPU 아키텍처에만 적용될 수 있습니다. 모든 아키텍처에 대해 빌드되지 않은 패키지 또는 모든 아키텍처에 영향을 미치지 않는 문제가 있을 수 있습니다. 모든 아키텍처에서 패키지를 사용할 수 있지만 문제가 하나에만 적용되는 경우 Amazon Linux는 일반적으로 영향을 받는 아키텍처만 참조하는 권고를 발행하지는 않았지만 이를 가정해서는 안 됩니다.

패키지 종속성의 특성으로 인해 권고에서 하나의 패키지를 참조하는 것이 일반적이지만, 해당 업데이트를 설치하려면 권고에 나열되지 않은 패키지를 포함한 다른 패키지 업데이트가 필요합니다. dnf 패키지 관리자가 필수 종속성 설치를 처리합니다.

권고 및 CVE

권고는 0개 이상의 CVE를 처리할 수 있으며 동일한 CVE를 참조하는 여러 권고가 있을 수 있습니다.

권고가 CVE를 참조할 수 있는 경우는 CVE가 아직(또는 어쩌다) 문제에 할당되지 않은 경우를 예로 들 수 있습니다.

CVE가 여러 패키지에 적용될 때 여러 권고가 동일한 CVE를 참조할 수 있는 예제입니다. 예를 들어 [CVE-2024-21208](#)은 Corretto 8, 11, 17 및 21에 적용됩니다. 이러한 각 Corretto 버전은 AL2023에서 별도의 패키지이며, 다음과 같이 각 패키지에 대한 권고가 있습니다. 즉 Corretto 8의 경우 [ALAS-2024-754](#), Corretto 11의 경우 [ALAS-2024-753](#), Corretto 17의 경우 [ALAS-2024-752](#), Corretto 21의 경우 [ALAS-2024-752](#)입니다. 이러한 Corretto 릴리스에는 모두 동일한 CVE 목록이 있지만 이를 가정해서는 안 됩니다.

패키지마다 특정 CVE를 다르게 평가할 수 있습니다. 예를 들어 특정 CVE가 심각도가 중요한 권고에서 참조되는 경우 심각도가 다른 동일한 CVE를 참조하는 다른 권고가 발행될 수 있습니다.

RPM 리포지토리 메타데이터는 각 권고에 대한 참조 목록을 허용합니다. Amazon Linux는 일반적으로 CVE만 참조하지만 메타데이터 형식은 다른 참조 유형을 허용합니다.

RPM 패키지 리포지토리 메타데이터는 사용 가능한 수정 사항이 있는 CVE만 참조합니다. [Amazon Linux 보안 센터 웹사이트의 탐색 섹션](#)에는 Amazon Linux가 평가한 CVE에 대한 정보가 포함되어 있습니다. 이 평가를 수행하면 다양한 Amazon Linux 릴리스 및 패키지에 대한 CVSS 기본 점수, 심각도 및 상태가 발생할 수 있습니다. 특정 Amazon Linux 릴리스 또는 패키지의 CVE 상태는 영향을 받지 않음, 수정 보류 중 또는 계획된 수정 없음일 수 있습니다. CVE의 상태 및 평가는 권고가 발표되기 전에 여러 번 어떤 식으로든 변경될 수 있습니다. 여기에는 Amazon Linux에 대한 CVE의 적용 가능성 재평가가 포함됩니다.

권고에서 참조하는 CVE 목록은 해당 권고를 처음 게시한 후 변경될 수 있습니다.

권고 텍스트

권고에는 권고를 생성한 이유인 문제 또는 문제를 설명하는 텍스트도 포함됩니다. 일반적으로 이 텍스트는 수정되지 않은 CVE 텍스트입니다. 이 텍스트는 Amazon Linux가 수정 사항을 적용한 패키지 버전과 다른 수정 사항을 사용할 수 있는 업스트림 버전 번호를 참조할 수 있습니다. 일반적으로 Amazon

Linux는 최신 업스트림 릴리스의 수정 사항을 백포트합니다. 권고 텍스트에서 Amazon Linux 버전으로 제공되는 버전과 다른 업스트림 릴리스를 언급하는 경우 권고의 Amazon Linux 패키지 버전은 Amazon Linux에 대해 정확합니다.

RPM 리포지토리 메타데이터의 권고 텍스트는 단순히 [Amazon Linux 보안 센터](#) 웹사이트에서 참조하는 자리 표시자 텍스트일 수 있습니다.

커널 라이브 패치 권고 사항

라이브 패치에 대한 권고는 권고가 적용되는 패키지(예: `kernel-livepatch-6.1.15-28.43`)와 다른 패키지(Linux 커널)를 참조한다는 점에서 고유합니다.

[커널 라이브 패치](#)에 대한 권고는 라이브 패치 패키지가 적용되는 특정 커널 버전에 대해 특정 라이브 패치 패키지가 해결할 수 있는 문제(예: CVE)를 참조합니다.

각 라이브 패치는 특정 커널 버전용입니다. CVE에 라이브 패치를 적용하려면 커널 버전에 적합한 라이브 패치 패키지를 설치하고 라이브 패치를 적용해야 합니다.

예를 들어 AL2023 커널 버전 6.1.56-82.125, 6.1.59-84.139 및 6.1.61-85.141에 대해 [CVE-2023-6111](#)을 라이브 패치할 수 있습니다. 이 CVE에 대한 수정 사항이 있는 새 커널 버전도 릴리스되었으며 [별도의 권고 사항](#)이 있습니다. AL2023에서 [CVE-2023-6111](#)을 처리하려면 [ALAS2023-2023-461](#)에서 지정한 커널 버전을 실행해야 하거나 이 CVE에 대한 라이브 패치가 있는 커널 버전 중 하나를 적용 가능한 라이브 패치가 적용된 상태로 실행해야 합니다.

이미 사용 가능한 라이브 패치가 있는 특정 커널 버전에 사용할 수 있는 새 라이브 패치가 있으면 `kernel-livepatch-KERNEL_VERSION` 패키지의 새 버전이 릴리스됩니다. 예를 들어, [ALASLIVEPATCH-2023-003](#) 권고는 세 개의 CVE를 포함하는 6.1.15-28.43 커널에 대한 라이브 패치가 포함된 `kernel-livepatch-6.1.15-28.43-1.0-1.amzn2023` 패키지와 함께 발행되었습니다. 나중에 [ALASLIVEPATCH-2023-009](#) 권고가 `kernel-livepatch-6.1.15-28.43-1.0-2.amzn2023` 패키지와 함께 발행되었습니다. 이는 다른 3개의 CVE에 대한 라이브 패치를 포함하는 6.1.15-28.43 커널용 이전 라이브 패치 패키지에 대한 업데이트였습니다. 또한 해당 특정 커널 버전에 대한 라이브 패치가 포함된 패키지와 함께 다른 커널 버전에 대한 다른 라이브 패치 권고 문제가 있었습니다.

커널 라이브 패치에 대한 자세한 정보는 [AL2023의 커널 라이브 패치](#) 섹션을 참조하세요.

보안 권고와 관련된 도구를 개발하는 사람은 [권고 및 updateinfo.xml에 대한 XML 스키마](#) 섹션에서 자세한 내용을 살펴보는 것이 좋습니다.

권고 및 `updateinfo.xml`에 대한 XML 스키마

`updateinfo.xml` 파일은 패키지 리포지토리 형식의 일부입니다. dnf 패키지 관리자가 [적용 가능한 권고 사항 나열](#) 및 [현재 위치에서 보안 업데이트 적용](#) 같은 기능을 구현하기 위해 구문 분석하는 메타데이터입니다.

리포지토리 메타데이터 형식을 구문 분석하기 위해 사용자 지정 코드를 작성하는 대신 dnf 패키지 관리자의 API를 사용하는 것이 좋습니다. AL2023의 dnf 버전은 AL2023 및 AL2 리포지토리 형식을 모두 구문 분석할 수 있으므로 API를 사용하여 OS 버전 중 하나에 대한 권고 정보를 검사할 수 있습니다.

[RPM 소프트웨어 관리](#) 프로젝트는 GitHub의 [rpm-metadata](#) 리포지토리에 RPM 메타데이터 형식을 문서화합니다.

`updateinfo.xml` 메타데이터를 직접 구문 분석하는 도구를 개발하는 경우 [rpm-metadata 설명서](#)에 주의하는 것이 좋습니다. 이 설명서에서는 메타데이터 형식의 규칙으로 합리적으로 해석할 수 있는 항목에 대한 많은 예외를 포함하여 와일드에서 관찰된 내용을 다룹니다.

GitHub의 [raw-historical-rpm-repository-examples](#) 리포지토리에는 점점 더 많은 실제 `updateinfo.xml` 파일 예제가 있습니다.

설명서에서 명확하지 않은 점이 있는 경우 GitHub 프로젝트에서 문제를 열어 질문에 답변하고 설명서를 적절하게 업데이트할 수 있습니다. 오픈 소스 프로젝트로서 풀 요청 업데이트 설명서도 환영합니다.

적용 가능한 권고 사항 나열

dnf 패키지 관리자는 어떤 권고 사항이 어떤 패키지 버전에서 수정되는지를 설명하는 메타데이터에 액세스할 수 있습니다. 따라서 인스턴스 또는 컨테이너 이미지에 적용할 수 있는 권고 사항을 나열할 수 있습니다.

Note

[AWS Systems Manager](#) 같은 도구는 이 기능을 사용하여 단일 인스턴스가 아닌 플릿 전체에서 관련된 업데이트를 표시할 수 있습니다.

업데이트를 나열할 때 특정 AL2023 릴리스의 메타데이터 또는 최신 릴리스의 메타데이터를 보도록 dnf에 지시할 수 있습니다.

Note

AL2023 릴리스가 생성되면 변경할 수 없습니다. 따라서 [Amazon Linux 보안 센터](#)의 신규 또는 업데이트된 권고는 AL2023의 신규 릴리스 메타데이터에만 추가됩니다.

이제 일부 AL2023 컨테이너 이미지에 적용되는 권고 사항을 살펴보는 예제를 살펴보겠습니다. 이러한 명령은 모두 EC2 인스턴스와 같은 컨테이너화되지 않은 환경에서 작동합니다.

Listing advisories in a specific version

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너 이미지에서 [2023.1.20230628](#) 릴리스의 권고 사항이 어떤 관련이 있는지 살펴보겠습니다.

Note

이 예제에서는 [2023.0.20230315](#) 및 [2023.1.20230628](#) 릴리스를 사용하며, 이는 AL2023의 최신 릴리스가 아닙니다. 최신 보안 업데이트가 포함된 최신 릴리스는 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.0.20230315](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 .0은 특정 릴리스의 이미지 버전을 나타냅니다. 이 이미지 버전은 일반적으로 0입니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성할 수 있으며, 컨테이너에 설치된 패키지와 관련된 권고 사항을 나열하도록 dnf에 요청할 것입니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.0.20230315.0
bash-5.2#
```

이제 `dnf updateinfo` 명령은 [2023.1.20230628](#) 릴리스의 어떤 권고 사항이 설치된 패키지와 관련이 있는지에 대한 요약을 표시하는 데 사용됩니다.

```
$ 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)
```

권고 목록을 가져오기 위해 `--list` 옵션을 `dnf updateinfo`에 제공할 수 있습니다.

```
$ 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

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너를 시작한 경우 latest AL2023 버전에서 사용할 수 있는 업데이트를 살펴보겠습니다. 작성 시점에 latest 릴리스는 [2023.5.20240708](#)이므로 이 예제에 나열된 업데이트는 해당 릴리스를 기준으로 합니다.

Note

이 예제에서는 [2023.4.20240319](#) 및 [2023.5.20240708](#) 릴리스를 사용하며, 후자는 작성 시점의 최신 릴리스입니다. 최신 릴리스에 대한 자세한 내용은 [AL2023 릴리스 노트](#)를 참조하세요.

이 예제에서는 [2023.4.20240319](#) 릴리스의 컨테이너 이미지로 시작합니다.

먼저 컨테이너 레지스트리에서 이 컨테이너 이미지를 가져옵니다. 끝에 있는 .1은 특정 릴리스의 이미지 버전을 나타냅니다. 이미지 버전은 일반적으로 0이지만 이 예제에서는 이미지 버전이 1인 릴리스를 사용합니다.

```
$ 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
```

이제 컨테이너 내에 셸을 생성하여 업데이트를 확인할 수 있습니다.

```
$ docker run -it public.ecr.aws/amazonlinux/amazonlinux:2023.4.20240319.1
bash-5.2#
```

dnf updateinfo 명령은 이제 설치된 패키지와 관련된 최신 릴리스의 권고 사항에 대한 요약을 표시하는 데 사용됩니다. 작성 당시 [2023.1.20230628](#)이 최신 릴리스였습니다.

```
$ 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)
```

권고 목록을 가져오기 위해 --list 옵션을 dnf updateinfo에 제공할 수 있습니다.

```
$ 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

```

현재 위치에서 보안 업데이트 적용

업데이트 적용에 대한 개요는 [DNF 및 리포지토리 버전에서 보안 업데이트](#) 섹션을 참조하세요. `dnf upgrade`에 대한 `--security` 옵션은 패키지 업데이트를 권고 사항이 있는 것으로만 제한합니다. 이 섹션의 나머지 부분에서는 특정 보안 업데이트만 설치하는 방법을 다룹니다.

Note

새 AL2023 릴리스에서 사용할 수 있는 모든 업데이트를 적용하는 것이 좋습니다. 보안 업데이트만 선택하거나 특정 업데이트만 선택하는 것은 규칙이 아닌 예외여야 합니다.

권고에 언급된 업데이트 적용

`dnf upgradeinfo` 출력의 첫 번째 열에 있는 권고 식별자를 사용하여 권고에 언급된 패키지에 대한 업데이트를 적용할 수 있습니다. `dnf` 패키지 관리자에게 권고의 패키지를 사용 가능한 최신 버전으로 업데이트하거나 권고에 언급된 버전으로만 업데이트하도록 지시할 수 있습니다. 업데이트가 이미 설치된 경우 업데이트 명령은 `no-op`입니다.

권고에 언급된 버전까지만 영향을 받는 패키지에 대한 업데이트를 적용하려면 `--advisory` 옵션을 사용하여 권고를 지정하면서 `dnf upgrade-minimal` 명령을 사용합니다. 다음 예제는 AL2023 버전 [2023.0.20230315](#) 컨테이너에서 `dnf upgrade-minimal`을 실행하고 있습니다.

```

$ 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!

```

권고를 해결하기 위해 필요한 최소 업데이트를 dnf가 수행하도록 하는 것이 요청이므로 --releasever=latest를 사용하더라도 동일한 패키지 버전이 업데이트됩니다.

--advisory 옵션과 함께 일반 dnf upgrade 명령을 사용하면 권고에 언급된 관련 패키지가 사용 가능한 최신 버전으로 업데이트되며, 이는 권고에 언급된 버전보다 최신일 수 있습니다.

Note

system-release 패키지가 업데이트되지 않는 한 dnf로 잠긴 AL2023 리포지토리의 버전은 변경되지 않습니다.

Warning

dnf로 잠긴 리포지토리의 버전을 변경하지 않고 AL2023의 다른 릴리스에서 업데이트를 설치하는 경우 후속 변경하는 dnf 작업에 주의를 기울여야 합니다. 예를 들어 패키지를 설치하거나 업데이트할 때 패키지 종속성이 최신 릴리스에서 변경되었을 수 있으므로 남아 있는 이전 릴리스는 이러한 새 종속성을 충족하지 못할 수 있습니다.

다음 예제는 AL2023 버전 [2023.0.20230315](#) 컨테이너에서 실행되며, 작성 시점에 AL2023의 최신 릴리스인 [2023.5.20240708](#)을 참조합니다. curl로 업데이트되는 버전이 update-minimal로 업데이트된 버전보다 최신이지만 이 최신 버전은 새로운 종속성을 가져옵니다.

```
$ 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!

```

AL2023의 SELinux 모드 설정

기본적으로 Security Enhanced Linux(SELinux)는 enabled 상태이며 AL2023의 경우 permissive 모드로 설정되어 있습니다. 허용 모드에서는 권한 거부 기록이 기록되지만 강제되지는 않습니다. SELinux는 커널의 주요 하위 시스템에 강력하고 유연한 필수 액세스 제어 (MAC) 아키텍처를 제공하는 커널 기능 및 유틸리티를 포함하고 있습니다.

SELinux는 기밀 유지 및 무결성 요구 사항에 따라 정보를 분리하도록 하는 강화된 메커니즘을 제공합니다. 이렇게 정보를 분리하면 애플리케이션 보안 메커니즘의 변조 및 우회 위협이 줄어듭니다. 또한 악의적이거나 결함이 있는 애플리케이션으로 인해 발생할 수 있는 피해도 제한합니다.

SELinux에는 일상적인 보안 목표를 충족하도록 설계된 샘플 보안 정책 구성 파일 세트가 포함되어 있습니다.

SELinux 기능에 대한 자세한 내용은 [SELinux 노트](#) 및 [정책 언어](#)를 참조하세요.

주제

- [AL2023의 기본 SELinux 상태 및 모드](#)
- [enforcing 모드로 변경](#)
- [AL2023용 SELinux 비활성화 옵션](#)

AL2023의 기본 SELinux 상태 및 모드

기본적으로 AL2023 SELinux는 enabled 상태이며 permissive 모드로 설정되어 있습니다. permissive 모드에서는 권한 거부 기록이 기록되지만 강제하지는 않습니다.

getenforce 또는 **sestatus** 명령은 현재 SELinux 상태, 정책 및 모드를 알려줍니다.

기본 상태가 enabled 및 permissive로 설정된 경우 **getenforce** 명령이 permissive로 변경됩니다.

다음 예제와 같이, 이 **sestatus** 명령으로 SELinux 상태와 현재 SELinux 정책이 반환됩니다.

```
$ 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
```

SELinux를 permissive 모드에서 실행하면 사용자가 파일에 레이블을 잘못 지정할 수 있습니다. disabled 상태에서 SELinux를 실행하면 파일에 레이블이 지정되지 않습니다. 잘못된 파일이나 레이블이 없는 파일 모두 enforcing 모드로 변경되면 문제를 일으킬 수 있습니다.

SELinux는 이 문제를 방지하기 위해 파일의 레이블을 자동으로 다시 지정합니다. SELinux는 상태를 enabled로 변경할 때 자동으로 레이블을 다시 지정하여 레이블 문제를 방지합니다.

enforcing 모드로 변경

enforcing 모드에서 SELinux를 실행하면 SELinux 유틸리티가 구성 정책을 enforcing합니다. SELinux는 정책의 규칙에 따라 액세스를 허용하거나 거부하여 일부 애플리케이션의 기능을 제어합니다.

현재 SELinux 모드를 찾으려면 getenforce 명령을 실행합니다.

```
getenforce
Permissive
```

구성 파일을 수정하여 **enforcing** 모드를 활성화합니다.

다음 단계에 따라 모드를 enforcing으로 변경할 수 있습니다.

1. /etc/selinux/config 파일을 수정하면 enforcing 모드로 변경됩니다. SELINUX 설정은 다음 예제와 같은 모습이어야 합니다.

```
SELINUX=enforcing
```

2. 시스템을 재시작하여 enforcing 모드 변경을 완료하세요.

```
$ sudo reboot
```

다음 부팅 시 SELinux는 시스템의 모든 파일 및 디렉터리에 레이블을 다시 지정합니다. SELinux는 SELinux가 disabled일 때 생성된 파일 및 디렉터리에 대한 SELinux 컨텍스트도 추가합니다.

enforcing 모드로 변경한 후 SELinux는 SELinux 정책 규칙이 잘못되었거나 누락되어 일부 작업을 거부할 수 있습니다. 다음 명령을 사용하여 SELinux가 거부한 작업을 볼 수 있습니다.

```
$ sudo ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts recent
```

cloud-init를 사용하여 **enforcing**을 활성화합니다.

또는 인스턴스를 실행할 때 다음 cloud-config을 사용자 데이터로 전송하여 enforcing 모드를 사용할 수 있습니다.

```
#cloud-config
selinux:
  mode: enforcing
```

이렇게 설정하면 인스턴스가 재부팅됩니다. 안정성을 높이려면 인스턴스를 재부팅하는 것이 좋습니다. 하지만 다음 cloud-config를 입력하면 재부팅을 건너뛸 수 있습니다.

```
#cloud-config
selinux:
  mode: enforcing
  selinux_no_reboot: 1
```

AL2023용 SELinux 비활성화 옵션

SELinux를 비활성화하면 SELinux 정책을 로드하거나 적용할 수 없으며 액세스 벡터 캐시(AVC) 메시지도 로깅되지 않습니다. SELinux를 실행하여 얻을 수 있는 모든 이점을 사용할 수 없습니다.

SELinux를 비활성화하는 대신 permissive 모드를 사용하는 것이 좋습니다. SELinux를 완전히 비활성화하는 것보다 permissive 모드에서 실행하는 비용이 약간 추가됩니다. SELinux를 비활성화한 후 enforcing 모드로 다시 전환하는 것보다 permissive 모드에서 enforcing 모드로 전환할 때 구성 수정이 훨씬 적습니다. 파일에 레이블을 지정할 수 있으며, 시스템에 활성 정책이 거부된 작업을 추적하고 기록할 수 있습니다.

SELinux를 **permissive** 모드로 변경

SELinux를 permissive 모드에서 실행하면 SELinux 정책을 적용할 수 없습니다. permissive 모드에서는 SELinux가 AVC 메시지를 로깅하지만 작업을 거부하지는 않습니다. 이러한 AVC 메시지를 문제 해결, 디버깅 및 SELinux 정책 개선에 사용할 수 있습니다.

SELinux를 허용 모드로 변경하려면 다음 단계를 사용합니다.

1. `/etc/selinux/config` 파일을 편집하여 `permissive` 모드로 변경합니다. SELINUX 값은 다음 예제와 같은 모습이어야 합니다.

```
SELINUX=permissive
```

2. 시스템을 다시 시작하여 `permissive` 모드 변경을 완료하세요.

```
sudo reboot
```

SELinux 비활성화

SELinux를 비활성화하면 SELinux 정책을 로드하거나 적용할 수 없으며 AVC 메시지도 로깅되지 않습니다. SELinux를 실행하여 얻을 수 있는 모든 이점을 사용할 수 없습니다.

SELinux를 비활성화하려면 다음 단계를 사용합니다.

1. `grubby` 패키지가 설치되어 있는지 확인합니다.

```
rpm -q grubby  
grubby-version
```

2. 커널 명령줄에 `selinux=0`을 추가하여 부트로더를 구성하세요.

```
sudo grubby --update-kernel ALL --args selinux=0
```

3. 시스템을 재시작합니다.

```
sudo reboot
```

4. `getenforce` 명령을 실행하여 SELinux가 Disabled 상태인지 확인합니다.

```
$ getenforce  
Disabled
```

SELinux에 대한 자세한 내용은 [SELinux 노트북](#) 및 [SELinux 구성](#) 섹션을 참조하세요.

AL2023에서 FIPS 모드 활성화

이 섹션에서는 AL2023에서 연방 정보 처리 표준(FIPS)을 활성화하는 방법을 안내합니다. FIPS에 대한 자세한 내용은 다음을 참조하세요.

- [연방 정보 처리 표준\(FIPS\)](#)
- [연방 정보 처리 표준 규정 준수 FAQs](#)

Note

이 섹션에서는 AL2023 FIPS 모드를 활성화하는 방법을 설명하며, AL2023 암호화 모듈의 인증 상태는 다루지 않습니다.

사전 조건

- 인터넷에 액세스하여 필수 패키지를 다운로드할 수 있는 기존 AL2023(AL2023.2 이상) Amazon EC2 인스턴스. AL2023 Amazon EC2 인스턴스를 실행하는 방법에 대한 자세한 내용은 [Amazon EC2 콘솔에서 AL2023 시작](#)을 참조하세요.
- SSH 또는 AWS Systems Manager를 사용하여 Amazon EC2 인스턴스를 연결합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 섹션을 참조하세요.

Important

ED25519 SSH 사용자 키는 FIPS 모드에서 지원되지 않습니다. ED25519 SSH 키 페어를 사용하여 Amazon EC2 인스턴스를 실행한 경우 다른 알고리즘(예: RSA)을 사용하여 새 키를 생성해야 합니다. 그렇지 않으면 FIPS 모드를 활성화한 후 인스턴스 액세스 권한을 잃을 수 있습니다. 자세한 내용은 Amazon EC2 사용 설명서의 [키 쌍 생성](#) 섹션을 참조하세요.

FIPS 모드 활성화

1. SSH 또는 AWS Systems Manager로 AL2023 인스턴스에 연결합니다.
2. 시스템이 최신 버전인지 확인합니다. 자세한 내용은 [AL2023에서 패키지 및 운영 체제 업데이트 관리](#) 섹션을 참조하세요.
3. `crypto-policies` 유틸리티가 설치되어 있고 최신 버전인지 확인합니다.

```
sudo dnf -y install crypto-policies crypto-policies-scripts
```

4. 다음 명령을 실행하여 FIPS 모드를 활성화합니다. 이렇게 하면 [AL2023 FAQ](#)에 나열된 모듈에 맞게 시스템 전체에서 FIPS 모드가 활성화됩니다.

```
sudo fips-mode-setup --enable
```

5. 다음 명령을 사용하여 인스턴스를 재부팅하세요.

```
sudo reboot
```

6. FIPS 모드가 활성화되었는지 확인하려면 인스턴스에 다시 연결하고 다음 명령을 실행합니다.

```
sudo fips-mode-setup --check
```

다음 예시 출력은 FIPS 모드가 활성화되었음을 나타냅니다.

```
FIPS mode is enabled.
```

AL2023 컨테이너에서 FIPS 모드 활성화

이 섹션에서는 AL2023 컨테이너에서 연방 정보 처리 표준(FIPS)을 활성화하는 방법을 안내합니다. FIPS에 대한 자세한 내용은 다음을 참조하세요.

- [연방 정보 처리 표준\(FIPS\)](#)
- [연방 정보 처리 표준 규정 준수 FAQs](#)

Note

이 섹션에서는 AL2023 컨테이너에서 FIPS 모드를 활성화하는 방법을 설명합니다. AL2023 암호화 모듈의 인증 상태는 다루지 않습니다.

사전 조건

- 인터넷에 액세스하여 필수 패키지를 다운로드할 수 있는 기존 AL2023(AL2023.2 이상) Amazon EC2 인스턴스. AL2023 Amazon EC2 인스턴스를 실행하는 방법에 대한 자세한 내용은 [Amazon EC2 콘솔에서 AL2023 시작](#)을 참조하세요.
- SSH 또는 AWS Systems Manager를 사용하여 Amazon EC2 인스턴스를 연결합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 섹션을 참조하세요.

Important

`fips-mode-setup` 명령은 컨테이너 내에서 올바르게 작동하지 않습니다. 아래 단계를 읽고 AL2023 컨테이너에서 FIPS 모드를 올바르게 구성하세요.

AL2023 컨테이너에서 FIPS 모드 활성화

1. FIPS 모드는 먼저 AL2023 컨테이너 호스트에서 활성화해야 합니다. [AL2023에서 FIPS 모드 활성화](#)의 지침에 따라 호스트에서 FIPS 모드를 활성화합니다.
2. SSH 또는 AWS Systems Manager를 사용하여 AL2023 컨테이너 호스트 인스턴스에 연결합니다.
3. AL2023 호스트가 FIPS 모드이고 `/proc/sys/crypto/fips_enabled`가 컨테이너 내에서 액세스할 수 있는 경우 AL2023 컨테이너에서 FIPS 모드가 자동으로 활성화됩니다. `/proc/sys/crypto/fips_enabled`의 내용이 0이면 FIPS가 활성화되지 않고 값 1은 FIPS 모드가 활성화되었음을 나타냅니다.

AL2023 호스트와 컨테이너 모두에서 다음 명령을 실행하여 FIPS가 활성화되었는지 확인할 수 있습니다.

```
cat /proc/sys/crypto/fips_enabled
```

4. 그런 다음 컨테이너 내에서 FIPS 암호화 정책을 활성화합니다. 이를 수행하는 방법에는 아래 옵션에 설명된 여러 가지가 있습니다. 해당 환경에 가장 적합한 옵션을 사용합니다.
 - a. `update-crypto-policies` 명령을 사용하여 컨테이너 내에서 수동으로 FIPS 암호화 정책을 활성화합니다.

```
# Run these commands inside the container
dnf install -y crypto-policies-scripts
update-crypto-policies --set FIPS
```

- b. AL2023 컨테이너 내에 bind 탑재를 생성합니다(다른 배포에서 podman이 작동하는 방식과 유사).

```
# 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 컨테이너가 AL2023 호스트의 암호화 정책과 일치하도록 바인드 탑재를 생성할 수도 있습니다. 다음 명령만 예제로 제공됩니다. 컨테이너와 호스트 간에 암호화 정책 및 패키지 버전에 호환되지 않는 차이가 있는 경우 이 구성으로 인해 문제가 발생할 수 있습니다.

```
sudo docker pull amazonlinux:2023
sudo docker run --mount type=bind,src=/etc/crypto-policies,dst=/etc/crypto-policies -it amazonlinux:2023
```

5. 위의 단계를 수행한 후 다음 명령을 사용하여 컨테이너에서 FIPS가 활성화되어 있는지 다시 확인할 수 있습니다.

```
$ cat /etc/crypto-policies/config
FIPS

$ cat /proc/sys/crypto/fips_enabled
1
```

AL2023에서 OpenSSL FIPS 공급자 전환

이 섹션에서는 AL2023에서 latest 및 certified OpenSSL FIPS 공급자 간에 전환하는 방법을 설명합니다.

FIPS에 대한 자세한 내용은 다음을 참조하세요.

- [연방 정보 처리 표준\(FIPS\)](#)
- [연방 정보 처리 표준 규정 준수 FAQs](#)
- [암호화 모듈 선택 및 사용에 대한 FedRAMP 정책](#)

⚠ Important

AL2023.7 이상에서 기본 OpenSSL FIPS 공급자는 정기적인 버그 수정 및 보안 업데이트를 수신하는 `openssl-fips-provider-latest` 패키지입니다.

아래 지침은 `openssl-fips-provider-certified` 패키지에 고정하려는 고객에게만 해당됩니다. 이 버전의 FIPS 공급자는 NIST 인증서의 체크섬과 일치하며 최신 업데이트가 없을 수 있습니다.

FIPS 인증 모듈 및 패키지 버전에 대한 자세한 내용은 [AL2023 FAQ](#) 섹션을 참조하세요.

사전 조건

- 인터넷에 액세스하여 필수 패키지를 다운로드할 수 있는 기존 AL2023(AL2023.7 이상) Amazon EC2 인스턴스. AL2023 Amazon EC2 인스턴스를 실행하는 방법에 대한 자세한 내용은 [Amazon EC2 콘솔에서 AL2023 시작](#)을 참조하세요.
- SSH 또는 AWS Systems Manager를 사용하여 Amazon EC2 인스턴스를 연결합니다. 자세한 내용은 [AL2023 인스턴스에 연결](#) 섹션을 참조하세요.
- AL2023에서 FIPS 모드를 활성화하려면 [AL2023에서 FIPS 모드 활성화](#)의 지침을 따릅니다.

openssl-fips-provider-latest 및 openssl-fips-provider-certified 중에서 전환

1. `dnf`를 사용하여 OpenSSL FIPS 공급자를 전환합니다.

```
sudo dnf -y swap openssl-fips-provider-latest openssl-fips-provider-certified
```

2. 인증된 OpenSSL FIPS 공급자를 사용하고 있는지 확인합니다. FIPS 모드의 AL2023에서 다음 명령을 실행합니다.

```
openssl list -providers
```

다음 결과가 표시됩니다.

```
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 커널 하드닝

6.1 Linux 커널은 AL2023의 여러 하드닝 옵션 및 기능으로 구성되고 빌드되었습니다.

커널 하드닝 옵션 (독립 아키텍처)

CONFIG 옵션	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	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
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_I	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음

CONFIG 옵션	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_CFI_PERMISSIVE</u>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_COMPAT</u>	y	y	y	y	y	y
<u>CONFIG_COMPAT_BRK</u>	n	n	n	n	n	n
<u>CONFIG_COMPAT_VDSO</u>	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n
<u>CONFIG_DEBUG_CREDENTIALS</u>	n	n	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_DEBUG_LIST</u>	y	y	y	y	y	y
<u>CONFIG_DEBUG_NOTIFIERS</u>	n	n	n	n	n	n
<u>CONFIG_DEBUG_SG</u>	n	n	n	n	n	n
<u>CONFIG_DEBUG_VIRTUAL</u>	n	n	n	n	n	n
<u>CONFIG_DEBUG_WX</u>	n	n	n	n	n	n

CONFIG 옵션	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_VMEM	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_DEFAULT_VMEM	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	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_HARDENED_USERCOPY_PAGESPAN	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음

CONFIG 옵션	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>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<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>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<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 옵션	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>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_KEXEC</u>	y	y	y	y	y	y
<u>CONFIG_KFENCE</u>	n	n	n	n	n	n
<u>CONFIG_LDISC_AUTOLOAD</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 옵션	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>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_PAGE_POISONING_ZERO</u>	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_PANIC_ON_OOPS</u>	y	y	y	y	y	y

CONFIG 옵션	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	n	n	n	n	n	n
CONFIG_RANDOMIZE_KERNEL_TBOOT_LOADER	y	y	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_RANDOMIZE_KERNEL_TCPU	y	y	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_REFCOUNT_FULL	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_SCHED_CORE	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
CONFIG_SCHED_STACK_END_CHECK	y	y	y	y	y	y

CONFIG 옵션	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_SECURITYLOCKDOWN_LSM</u>	y	y	y	y	y	y
<u>CONFIG_SECURITYLOCKDOWN_LSM_EARLY</u>	y	y	y	y	y	y
<u>CONFIG_SECURITYSELINUX_BOOTPARAM</u>	y	y	y	y	y	y

CONFIG 옵션	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	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_SECURITY_WRITABLE_HOOKS	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
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 옵션	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_SLUB_DEBUG</u>	y	y	y	y	y	y
<u>CONFIG_STACKPROTECTOR</u>	y	y	y	y	y	y
<u>CONFIG_STACKPROTECTOR_STRONG</u>	y	y	y	y	y	y
<u>CONFIG_STACKTRACE_R</u>	n	n	n	n	n	n
<u>CONFIG_STACKTRACE_DEV</u>	n	n	n	n	n	n
<u>CONFIG_STACKTRACE_KERNEL_RWX</u>	y	y	y	y	y	y
<u>CONFIG_STACKTRACE_MODULE_RWX</u>	y	y	y	y	y	y
<u>CONFIG_SYSCALL_COOKIES</u>	y	y	y	y	y	y
<u>CONFIG_VMAP_STACK</u>	y	y	y	y	y	y

CONFIG 옵션	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

런타임 시 ACPI 메서드를 삽입/교체 가능 (CONFIG_ACPI_CUSTOM_METHOD)

root 사용자가 임의의 커널 메모리에 쓸 수 있으므로 Amazon Linux에서 이 옵션은 비활성화됩니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

기타 바이너리 포맷 (binfmt_misc)

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이는 AL2023 선택 기능이며 커널 모듈로 빌드됩니다.

BUG() 지원

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

커널 메모리 구조의 유효성을 검사할 때 커널에서 데이터 손상이 발생하면 **BUG()**가 발생합니다.

Linux 커널 기능으로 데이터 구조의 내부 일관성을 검사할 수 있으며 데이터 손상이 감지되면 **BUG()**가 발생합니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

COMPAT_BRK

이 옵션을 비활성화(Amazon Linux에서 커널 구성)하면 `randomize_va_space` sysctl 설정은 2로 기본 설정되며 mmap 베이스, 스택 및 VDSO 페이지 무작위화를 기반으로 힙 무작위화도 활성화됩니다.

이 옵션은 1996년 이전의 `libc.so.5` 바이너리와 호환하기 하기 위해 커널에 존재합니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

COMPAT_VDSO

이 구성 옵션은 aarch64가 아닌 x86-64에 해당합니다. n로 설정하면 Amazon Linux 커널에서 32비트 가상 동적 공유 객체(VDSO)를 예측 가능한 주소에 표시할 수 없습니다. 이 옵션을 n로 설정하면 glibc 오류가 발생한 것으로 알려진 가장 최신 버전은 2004년에 출시된 glibc 2.3.3입니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

CONFIG_DEBUG 게이트 강화

CONFIG_DEBUG가 제어하는 Linux 커널 구성 옵션은 일반적으로 디버깅 문제를 위해 빌드된 커널에서 사용하려고 만들었으며 성능은 우선 순위가 아닙니다. AL2023은 CONFIG_DEBUG_LIST 하드닝 옵션을 활성화합니다.

IOMMU를 구성하기 전에 EFI 스텝에서 PCI 디바이스용 DMA를 비활성화하세요.

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

커널과 사용자 공간 간에 메모리를 복사하기 위한 강화

커널이 사용자 공간으로 메모리를 복사하거나 사용자 공간 메모리를 복사해야 하는 경우 이 옵션을 사용하면 일부 검사 기능을 통해 일부 클래스의 힙 오버플로 문제를 방지할 수 있습니다.

커널 4.16~5.15에는 커널 개발자가 WARN()를 통해 누락된 허용 목록 항목을 찾을 수 있도록 하는 CONFIG_HARDENED_USERCOPY_FALLBACK 옵션이 있었습니다. AL2023에서 6.1 커널을 제공하므로 이 옵션은 더 이상 AL2023이 필요하지 않습니다.

이 CONFIG_HARDENED_USERCOPY_PAGESPAN 옵션은 주로 개발자를 위한 디버깅 옵션으로 커널에 존재했으며 AL2023 6.1 커널에는 더 이상 없습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

하이버네이트 지원

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이 옵션을 활성화해야 [온디맨드 인스턴스 하이버네이트](#)와 [중단된 스팟 인스턴스 하이버네이트](#) 기능을 지원할 수 있습니다.

난수 생성

AL2023 커널은 EC2에서 사용할 수 있는 적절한 엔트로피를 보장합니다.

CONFIG_INET_DIAG

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이는 AL2023 선택 기능이며 커널 모듈로 빌드됩니다.

할당 및 할당 해제 시 모든 커널 페이지 및 슬래브 할당자 메모리 0

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 기본 기능으로 설정하면 성능에 영향을 미칠 수 있으므로 AL2023에서는 이러한 옵션이 비활성화됩니다. 커널 명령줄에 `init_on_alloc=1`을 추가하면 `CONFIG_INIT_ON_ALLOC_DEFAULT_ON` 동작을 활성화할 수 있고, 추가하여 `init_on_free=1`를 추가하면 `CONFIG_INIT_ON_FREE_DEFAULT_ON` 동작을 활성화할 수 있습니다.

모든 스택 변수를 0(`CONFIG_INIT_STACK_ALL_ZERO`)으로 초기화합니다.

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이 옵션은 GCC 12 이상에서 작동하지만 AL2023에 GCC 11을 함께 제공합니다.

커널 모듈 서명

AL2023은 커널 모듈에 서명하고 서명을 검증합니다. 모듈에 유효한 서명이 있어야 하는 이 `CONFIG_MODULE_SIG_FORCE` 옵션은 타사 모듈을 구축하는 사용자의 호환성을 유지하기 위해 활성화되지 않습니다. 모든 커널 모듈에 서명이 있는지 확인하고 싶은 사용자는 이것이 적용된 [Linux 보안 모듈\(LSM\) 잠금](#)을 구성할 수 있습니다.

kexec

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이 옵션은 `kdump` 기능을 사용할 수 있도록 활성화되어 있습니다.

IOMMU 지원

AL2023은 IOMMU 지원을 활성화합니다. 이 `CONFIG_IOMMU_DEFAULT_DMA_STRICT` 옵션은 기본적으로 활성화되어 있지 않지만 커널 명령줄에 `iommu.passthrough=0 iommu.strict=1`를 추가하면 이 기능을 설치할 수 있습니다.

kfence

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

레거시 **pty** 지원

AL2023은 최신 PTY 인터페이스(devpts)를 사용합니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

Linux 보안 모듈(LSM) 잠금

AL2023에서 lockdown LSM을 빌드할 수 있으며, 보안 부팅을 사용할 때 LSM으로 커널을 잠급니다.

CONFIG_LOCK_DOWN_KERNEL_FORCE_CONFIDENTIALITY 옵션이 활성화되어 있지 않습니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 보안 부팅을 사용하지 않는 경우 Lockdown LSM을 활성화하고 원하는 대로 구성할 수 있습니다.

페이지 중독

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. [할당 및 할당 해제 시 모든 커널 페이지 및 슬래브 할당자 메모리 0](#)와 마찬가지로 AL2023 커널에서는 성능에 영향을 미칠 수 있으므로 비활성화되어 있습니다.

스택 프로텍터

AL2023 커널에 -fstack-protector-strong 옵션이 있는 GCC 스택 프로텍터 기능을 빌드했습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

seccomp BPF API

seccomp 하드닝 기능으로 systemd와 컨테이너 런타임과 같은 소프트웨어에서 사용자 공간 애플리케이션을 하드닝할 수 있습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

panic() 타임아웃

AL2023 커널에서 이 값은 0으로 설정됩니다. 즉, 패닉이 발생한 후 커널이 재부팅되지 않습니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSP\) 권장 설정](#) 중 하나지만 KSP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이는 sysctl, /proc/sys/kernel/panic 및 커널 명령줄로 구성할 수 있습니다.

보안 모델

AL2023에서 SELinux 허용 모드가 활성화되어 있습니다. 자세한 내용은 [AL2023의 SELinux 모드 설정](#) 단원을 참조하십시오.

[Linux 보안 모듈\(LSM\) 잠금](#) 및 yama 모듈도 활성화되어 있습니다.

/proc/kcore

이 옵션은 [커널 자체 보호 프로젝트 \(KSP\) 권장 설정](#) 중 하나지만 KSP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

시스템 호출 입력 시 커널 스택 오프셋 랜덤

이 옵션은 [커널 자체 보호 프로젝트 \(KSP\) 권장 설정](#) 중 하나지만 KSP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 커널 명령줄을 randomize_kstack_offset=on로 설정하면 이 기능을 활성화할 수 있습니다.

참조 카운팅 검사 (CONFIG_REFCOUNT_FULL)

이 옵션은 [커널 자체 보호 프로젝트 \(KSP\) 권장 설정](#) 중 하나지만 KSP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. 이 옵션은 성능에 영향을 미칠 수 있으므로 현재 활성화되어 있지 않습니다.

스케줄러의 SMT 코어 인식 (CONFIG_SCHED_CORE)

AL2023 커널은 CONFIG_SCHED_CORE로 빌드되었으며 사용자 공간 애플리케이션에서 prctl(PR_SCHED_CORE)을 사용할 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

schedule()(CONFIG_SCHED_STACK_END_CHECK) 호출 시 스택 손상 확인

AL2023 커널은 CONFIG_SCHED_STACK_END_CHECK가 활성화된 상태로 빌드되었습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

메모리 할당기 강화

AL2023 커널을 사용하면 CONFIG_SHUFFLE_PAGE_ALLOCATOR, CONFIG_SLAB_FREELIST_HARDENED, CONFIG_SLAB_FREELIST_RANDOM 옵션으로 커널 메모리 할당을 하드닝할 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

SLUB 디버깅 지원

AL2023 커널에서 커널 명령줄로 할당자의 디버깅 기능을 활성화할 수 있는 옵션 CONFIG_SLUB_DEBUG를 사용할 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

CONFIG_STATIC_USERMODEHELPER

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다. CONFIG_STATIC_USERMODEHELPER를 사용하려면 배포판의 특별 지원이 필요한데 현재 Amazon Linux에는 없습니다.

읽기 전용 커널 텍스트 및 rodata (CONFIG_STRICT_KERNEL_RWX와 CONFIG_STRICT_MODULE_RWX)

AL2023 커널은 커널, 커널 모듈 텍스트, rodata 메모리를 읽기 전용이지만 텍스트가 아닌 메모리를 실행 불가능 상태로 표시합니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

TCP syncookie 지원 (CONFIG_SYN_COOKIES)

AL2023 커널은 TCP syncookies 지원을 통해 빌드됩니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

가드 페이지가 있는 가상 매핑 스택 (CONFIG_VMAP_STACK)

AL2023 커널은 CONFIG_VMAP_STACK으로 빌드되어 가드 페이지가 있는 가상 매핑 커널 스택을 사용할 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

컴파일러 경고를 오류(CONFIG_WERROR)로 빌드

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

함수를 종료(CONFIG_ZERO_CALL_USED_REGS)할 때 레지스터 제로화

이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

사용자 공간 할당에 필요한 최소 주소

이 하드닝 옵션으로 커널 NULL 포인터 버그의 영향을 줄일 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

clang 특정 강화 옵션

AL2023 커널은 GCC 대신 clang으로 빌드되었으므로 CONFIG_CFI_CLANG 하드닝 옵션을 활성화할 수 없으며 CONFIG_CFI_PERMISSIVE 역시 사용할 수 없습니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

x86-64 전용 커널 하드닝 옵션

CONFIG 옵션	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	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
CONFIG_AMD_IOMMU_V2	해당 사항 없음	y	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_IA32_EMULATION	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
CONFIG_INTEL_IOMMU	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
CONFIG_INTEL_IOMMU	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n

CONFIG 옵션	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>__DEFAULT__</u> <u>ON</u>						
<u>CONFIG_INET_IOMMU_SVM</u>	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n
<u>CONFIG_LEGACY_VSYSCALL_NONE</u>	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n
<u>CONFIG_MODIFY_LDT_SYSCALL</u>	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n
<u>CONFIG_PAGE_TABLE_ISOLATION</u>	해당 사항 없음	y	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
<u>CONFIG_RANDOMIZE_MEMORY</u>	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
<u>CONFIG_X86_64</u>	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
<u>CONFIG_X86_64_MSR</u>	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y
<u>CONFIG_X86_64_VSYSCALL_EMULATION</u>	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음	y

CONFIG 옵션	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_X86_X32	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_X86_X32_ABI	해당 사항 없음	n	해당 사항 없음	n	해당 사항 없음	n

x86-64 지원

물리적 주소 확장(PAE) 및 실행 안 함(NX) 비트 지원이 x86-64 기본 지원입니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

AMD 및 Intel IOMMU 지원

AL2023 커널은 AMD와 Intel IOMMUs를 지원합니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

이 CONFIG_INTEL_IOMMU_DEFAULT_ON 옵션은 설정되지 않았지만 커널 명령줄로 intel_iommu=on를 작성하면 활성화됩니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

이 CONFIG_INTEL_IOMMU_SVM 옵션은 현재 AL2023에 활성화되어 있지 않습니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

32비트 사용자 공간 지원

Important

32비트 x86 사용자 공간 지원은 중단되었으며 향후 Amazon Linux 주요 버전에서 32비트 사용자 공간 바이너리를 실행할 수 없습니다.

Note

AL2023에는 더 이상 32비트 패키지가 포함되지 않지만, 커널은 32비트 사용자 공간 실행을 계속 지원합니다. 자세한 정보는 [32비트 x86 \(i686\) 패키지](#)를 참조하세요.

32비트 사용자 공간 애플리케이션 실행하려면 AL2023에서 CONFIG_X86_VSYSCALL_EMULATION 옵션이 아닌 CONFIG_IA32_EMULATION, CONFIG_COMPAT, CONFIG_X86_VSYSCALL_EMULATION 옵션을 활성화합니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

64비트 프로세서 x32 네이티브 32비트 ABI는 활성화되지 않았습니다 (CONFIG_X86_X32 및 CONFIG_X86_X32_ABI). 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

x86 모델별 레지스터(MSR) 지원

이 CONFIG_X86_MSR 옵션은 turbostat 지원을 위해 활성화되어 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 \(KSPP\) 권장 설정](#) 중 하나지만 KSPP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

modify_ldt 시스콜

AL2023 사용자 프로그램에서 modify_ldt syscall을 사용하여 x86 로컬 디스크립터 테이블(LDT)을 수정할 수 없습니다. 이 호출을 사용하려면 16비트 또는 세그먼트 코드를 실행해야 하며 이 호출이 없으면 dosemu 소프트웨어, WINE에서 실행하는 프로그램, 실행하거나 아주 오래된 스테딩 라이브러리가 중단됩니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

사용자 모드에서 커널 매핑을 제거합니다.

AL2023에서 대부분의 커널 주소가 사용자 공간에 매핑되지 않도록 커널이 구성되어 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

커널 메모리 섹션 무작위화

AL2023에서 커널 메모리 섹션의 기본 가상 주소를 무작위로 지정하도록 커널을 구성할 수 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

aarch64 전용 커널 하드닝 옵션

CONFIG 옵션	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	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음
CONFIG_ARM64_BTI_KERNEL	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음	해당 사항 없음
CONFIG_ARM64_PTR_AUTH	y	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음
CONFIG_ARM64_PTR_AUTH_KERNEL	y	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음
CONFIG_ARM64_SW_TTBR0_PAN	y	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음
CONFIG_UNMAP_KERNEL_AT_EL0	y	해당 사항 없음	y	해당 사항 없음	y	해당 사항 없음

브랜치 타겟 식별

AL2023 커널에 브랜치 타겟 식별(CONFIG_ARM64_BTI) 기능이 있습니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

이 CONFIG_ARM64_BTI_KERNEL 옵션은 GCC로 빌드되어 AL2023에서 사용할 수 없으며, 이 옵션으로 커널 빌드하기는 [gcc 버그로 인해 현재 업스트림 커널에서 비활성화되어 있습니다](#). 이 옵션은 [커널 자체 보호 프로젝트 \(KSP\) 권장 설정](#) 중 하나지만 KSP 권장 설정으로 AL2023 구성 옵션을 설정하지 않습니다.

포인터 인증 (CONFIG_ARM64_PTR_AUTH)

AL2023 커널은 반환 지향 프로그래밍(ROP) 기술을 완화하는 데 사용할 수 있는 포인터 인증 확장(ARMv8.3 확장 프로그램 일부)을 지원합니다. [Graviton](#) 포인터 인증에 필요한 하드웨어 지원은 Graviton 3에 추가되었습니다.

이 CONFIG_ARM64_PTR_AUTH 옵션은 활성화되어 있으며 사용자 공간 포인터 인증을 지원합니다. 이 CONFIG_ARM64_PTR_AUTH_KERNEL 옵션도 활성화되었으므로 AL2023 커널에서 자체적으로 반환 주소 보호 기능을 사용할 수 있습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

권한 있는 액세스 에뮬레이션 (TTBR0_EL1 전환 사용하지 않음)

사용자 액세스 루틴으로만 일시적으로만 TTBR0_EL1이 유효한 값으로 설정되므로 이 옵션으로 커널이 사용자 공간 메모리에 직접 액세스하는 것을 방지할 수 있습니다.

이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

사용자 공간에서 실행 중인 경우 커널 매핑을 해제합니다.

AL2023 커널이 사용자 공간(CONFIG_UNMAP_KERNEL_AT_EL0)에서 실행될 때 커널 매핑이 해제됩니다. 이 옵션은 [커널 자체 보호 프로젝트 권장 설정](#) 중 하나입니다.

AL2023에서 리포지토리 메타데이터 서명

릴리스부터 AL2023 2023.11.20260406 리포지토리에는 리포지토리 메타데이터에 대한 암호화 서명이 포함됩니다. 각 리포지토리의 repomd.xml 파일에는 패키지를 다운로드하기 전에 리포지토리 메타데이터의 신뢰성과 무결성을 확인하는 데 사용할 수 있는 분리된 GPG 서명 파일 (repomd.xml.asc)이 함께 제공됩니다.

이 서명은 개별 패키지를 확인하는 기존 RPM 패키지 서명(gpgcheck)에 추가됩니다. 리포지토리 메타데이터 서명은 사용 가능한 패키지 목록 및 체크섬과 같은 리포지토리의 콘텐츠를 설명하는 메타데이터를 확인합니다.

리포지토리 메타데이터 서명 작동 방식

AL2023 리포지토리가 게시되면 AWS KMS 키를 사용하여 리포지토리 메타데이터(repomd.xml)에 서명됩니다. 결과 분리된 서명(repomd.xml.asc)은 리포지토리의 메타데이터와 함께 배치됩니다.

리포지토리 구성 `repo_gpgcheck`에서 활성화하면 리포지토리 메타데이터를 사용하기 전에 GPG 퍼블릭 키에 대해 `repomd.xml.asc` 서명을 DNF 자동으로 다운로드하고 확인합니다. 서명 확인에 실패하면 리포지토리 메타데이터를 DNF 거부하고 해당 리포지토리의 패키지 작업을 진행하지 않습니다. 에 대한 자세한 내용은 [DNF 구성](#) `repo_gpgcheck`참조를 참조하세요.

다음 AL2023 리포지토리에는 서명된 메타데이터가 포함됩니다.

- 코어 리포지토리(`amazonlinux`)
- 커널 Livepatch 리포지토리(`kernel-livepatch`)
- NVIDIA 리포지토리(`amazonlinux-nvidia`)
- Amazon Linux 리포지토리용 보조 패키지(`amazonlinux-spall`)

gpgcheck와의 차이점 `repo_gpgcheck`

설정	확인 항목	AL2023의 기본값
<code>gpgcheck=1</code>	설치 전에 개별 RPM 패키지의 GPG 서명을 확인합니다.	활성화됨
<code>repo_gpgcheck=1</code>	리포지토리를 사용하기 전에 리포지토리 메타데이터(<code>repomd.xml</code>)의 GPG 서명을 확인합니다.	비활성화됨 (2023.12분기별 릴리스부터 기본적으로 활성화됨)

`gpgcheck` 및 `repo_gpgcheck` 모두 활성화하는 것이 좋습니다. 이렇게 하면 리포지토리 메타데이터와 개별 패키지가 모두 사용 전에 확인됩니다.

리포지토리 메타데이터 확인 활성화

구성 파일을 업데이트하여 개별 리포지토리에 대한 리포지토리 메타데이터 확인을 활성화할 수 있습니다.

Important

2023.12 분기별 릴리스부터 `repo_gpgcheck=1`은 AL2023 리포지토리 구성 파일에서 기본적으로 활성화됩니다.

특정 리포지토리에 대해 활성화

의 AL2023 리포지토리 구성 파일은 `repo_gpgcheck=0` 기본적으로 `/etc/yum.repos.d/` 설정됩니다. 리포지토리 메타데이터 확인을 활성화하려면 리포지토리 구성1에서이 값을 `1`로 변경합니다. 예를 들어 코어 리포지토리에 대해 활성화하려면

```
[amazonlinux]
name=Amazon Linux 2023 repository
...
gpgcheck=1
repo_gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-amazon-linux-2023
```

리포지토리 메타데이터 서명이 작동하는지 확인

를 활성화 `repo_gpgcheck=1`한 후 DNF 캐시를 지우고 메타데이터를 새로 고쳐 메타데이터 확인이 작동하는지 확인할 수 있습니다.

```
[ec2-user ~]$ sudo dnf clean metadata
[ec2-user ~]$ sudo dnf makecache
```

메타데이터 확인에 성공하면는 GPG 키(아직 가져오지 않은 경우)를 DNF 가져오고 오류 없이 메타데이터 캐시를 생성합니다. 출력은 다음과 비슷합니다.

```
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.
```

서명 확인에 실패하면는 GPG 서명 확인 실패 및 메타데이터 캐시 생성 실패를 나타내는 오류 메시지를 DNF 표시합니다.

AL2023 리포지토리를 GPG 퍼블릭 키

리포지토리 메타데이터 확인에 사용되는 GPG 퍼블릭 키는 해당 리포지토리 구성 RPMs에서 설치합니다 `/etc/pki/rpm-gpg/`. 다음 표에는 각 리포지토리에서 사용하는 퍼블릭 키가 나열되어 있습니다.

리포지토리	패키지 서명 키	Repdata 서명 키	에 배포됨
코어 (amazonlinux)	RPM-GPG-KEY-amazon -linux-2023	RPM-GPG-KEY-amazon -linux-2023	system-release
커널 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

이러한 키는 해당 리포지토리 구성 RPM을 설치할 때 자동으로 설치됩니다.

AL2023의 UEFI 보안 부팅

AL2023은 릴리스 2023.1부터 UEFI 보안 부팅을 지원합니다. UEFI와 UEFI 보안 부팅을 모두 지원하는 Amazon EC2 인스턴스로 AL2023을 사용하세요. 자세한 내용은 [Amazon EC2 사용 설명서](#)의 UEFI 부팅 모드에서 Amazon EC2 인스턴스를 시작하기 위한 요구 사항 섹션을 참조하세요.

UEFI 보안 부팅이 활성화된 AL2023 인스턴스는 Linux 커널과 모듈을 포함하여 Amazon이 서명한 커널 수준 코드만 허용하므로 인스턴스가 AWS가 서명한 커널 수준 코드만 실행하도록 보장할 수 있습니다.

Amazon EC2 인스턴스와 UEFI 보안 부팅에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 인스턴스용 UEFI 보안 부팅](#) 섹션을 참조하세요.

사전 조건

- AL2023 릴리스 2023.1 이상에 설치된 AMI가 필요합니다.
- 인스턴스 유형은 UEFI 보안 부팅을 지원해야 합니다. 자세한 내용은 [Amazon EC2 사용 설명서](#)의 UEFI 부팅 모드에서 Amazon EC2 인스턴스를 시작하기 위한 요구 사항 섹션을 참조하세요.

AL2023에서 UEFI 보안 부팅 활성화

기본 AL2023 AMI는 당사 키 서명이 있는 커널과 부트로더와 통합되어 있습니다. 기존 인스턴스를 등록하거나 스냅샷 이미지 등록으로 UEFI 보안 부팅이 활성화된 AMI를 생성하여 UEFI 보안 부팅을 사용할 수 있습니다. UEFI 보안 부팅은 기본 AL2023 AMI에서 사용할 수 없습니다.

인스턴스 유형이 UEFI를 지원하는 경우 이러한 AMI로 시작된 인스턴스가 UEFI 펌웨어를 사용하도록 AL2023 AMI 부팅 모드는 `uefi-preferred`로 설정되어 있습니다. 인스턴스 유형이 UEFI를 지원하지 않는 경우 인스턴스는 레거시 BIOS 펌웨어에서 실행합니다. 인스턴스가 레거시 BIOS 모드에서 시작되면 UEFI 보안 부팅이 되지 않습니다.

Amazon EC2 인스턴스의 AMI 부팅 모드에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 부팅 모드를 사용한 인스턴스 시작 동작](#) 섹션을 참조하세요.

주제

- [기존 인스턴스 등록](#)
- [스냅샷 이미지 등록](#)
- [패키지 업데이트](#)
- [AL2023에서 UEFI 보안 부팅 작동 방식](#)
- [자체 키 등록](#)

기존 인스턴스 등록

기존 인스턴스를 등록하려면 펌웨어가 부트로더를 검증하고 부트로더가 다음 부팅 시 커널을 검증하는 키 세트를 특정 UEFI 펌웨어 변수로 입력하세요.

1. Amazon Linux는 등록 프로세스를 간소화하는 도구를 제공합니다. 다음 명령을 실행하여 필요한 키와 인증서 세트를 인스턴스에 프로비저닝합니다.

```
sudo amazon-linux-sb enroll
```

2. 다음 명령을 실행하여 인스턴스를 재부팅합니다. 인스턴스가 재부팅되면 UEFI 보안 부팅이 활성화됩니다.

```
sudo reboot
```

Note

Amazon Linux AMI는 현재 Nitro 신뢰 플랫폼 모듈(NitroTPM)을 지원하지 않습니다. UEFI 보안 부팅뿐만 아니라 NitroTPM이 필요한 경우 다음 섹션의 정보를 사용하세요.

스냅샷 이미지 등록

Amazon EC2 `register-image` API를 사용하여 Amazon EBS 루트 볼륨의 스냅샷에서 AMI를 등록할 때 UEFI 변수 저장소의 상태가 포함된 바이너리 볼륨으로 AMI를 프로비저닝할 수 있습니다. AL2023 UefiData로 UEFI 보안 부팅이 활성화되므로 이전 섹션의 단계를 실행하지 않아도 됩니다.

바이너리 볼륨 생성 및 사용에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [미리 채워진 변수 저장소가 있는 바이너리 볼륨 생성](#) 섹션을 참조하세요.

AL2023은 Amazon EC2 인스턴스에서 직접 사용할 수 있는 사전 빌드된 바이너리 볼륨을 제공합니다. 바이너리 볼륨은 실행 중인 인스턴스의 `/usr/share/amazon-linux-sb-keys/uefi.vars`에 있습니다. AL2023 AMI 2023.1 릴리스부터 기본으로 설치된 `amazon-linux-sb-keys` RPM 패키지에 이 볼륨이 있습니다.

Note

최신 버전의 키와 폐기를 사용하려면 AMI를 생성할 때 사용한 것과 동일한 AL2023 릴리스의 볼륨을 사용하세요.

이미지를 등록할 때는 `uefi`로 설정된 [RegisterImage](#) API의 `BootMode` 파라미터를 사용하는 것이 좋습니다. 이렇게 하면 `TpmSupport` 파라미터를 `v2.0`로 설정하여 NitroTPM을 활성화할 수 있습니다. UEFI를 지원하지 않는 인스턴스 유형으로 전환할 때 UEFI 보안 부팅이 활성화되고 실수로 비활성화되지 않도록 `BootMode` 설정을 `uefi`로 하세요.

NitroTPM에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 인스턴스용 NitroTPM](#) 섹션을 참조하세요.

폐기 업데이트

Amazon Linux에서 업데이트된 키 서명이 포함된 최신 부트로더 버전 `grub2` 또는 Linux 커널을 배포해야 합니다. 이 경우 구 버전 부트로더에서 악용될 수 있는 버그가 UEFI 보안 부팅 검증 프로세스를 우회할 가능성을 방지하기 위해 구 버전 키를 폐기해야 합니다.

grub2 또는 kernel 패키지를 업데이트하면 실행 중인 인스턴스의 UEFI 변수 저장소 취소 목록을 자동으로 업데이트합니다. 즉, UEFI 보안 부팅을 활성화하면 패키지 보안 업데이트를 설치한 후에는 이전 패키지 버전을 더 이상 실행할 수 없습니다.

AL2023에서 UEFI 보안 부팅 작동 방식

다른 Linux 배포판과 달리 Amazon Linux에는 1단계 부트로더 역할을 하는 shim (shim) 추가 구성 요소가 없습니다. shim은 일반적으로 Microsoft 키 서명이 있습니다. 예를 들어 shim이 포함된 Linux 배포판에서 shim은 자체 코드를 사용하여 Linux 커널을 검증하는 grub2 부트로더를 로드합니다. 또한 shim은 mokutil 도구로 UEFI 변수 저장소에서 머신 소유 키(MOK) 데이터베이스에 자체 키 및 폐기 세트를 유지 관리합니다.

Amazon Linux에는 shim이 없습니다. AMI 소유자가 UEFI 변수를 관리하기 때문에 이 중간 단계는 필요하지 않으며 시작 및 부팅에 부정적인 영향을 미칠 수 있습니다. 또한 원하지 않는 바이너리가 실행될 가능성을 줄이기 위해 기본적으로 공급업체 키에 대한 신뢰를 포함시키지 않았습니다. 하지만 고객은 원한다면 바이너리를 포함시킬 수 있습니다.

Amazon Linux를 사용하면 UEFI로 grub2 부트로더를 로드하고 검증할 수 있습니다. Linux 커널을 로드한 후 UEFI를 사용하여 Linux 커널을 검증하도록 grub2 부트로더를 수정했습니다. 따라서 일반 UEFI db 변수(인증된 키 데이터베이스)에 저장된 동일한 인증서를 사용하여 Linux 커널을 검증하고 부트로더 및 다른 UEFI 바이너리와 동일한 dbx 변수(폐기 데이터베이스)를 테스트합니다. db 데이터베이스 및 dbx 데이터베이스에 대한 액세스를 제어하는 자체 PK 및 KEK 키로 shim과 같은 중개자 없이 필요에 따라 서명된 업데이트 및 폐기를 수행할 수 있습니다.

UEFI 보안 부팅에 대한 자세한 내용은 Amazon EC2 사용 설명서의 [Amazon EC2 인스턴스에서 UEFI 보안 부팅이 작동하는 방식](#) 섹션을 참조하세요.

자체 키 등록

이전 섹션에서 설명했듯이 Amazon EC2에서 UEFI 보안 부팅을 사용하기 위해 shimAmazon Linux에 가 있어도 됩니다. 다른 Linux 배포판에 대한 설명서에서 AL2023에는 없는 mokutil로 머신 소유자 키(MOK) 데이터베이스를 관리하는 방법에 대한 문서가 있습니다. shim 및 MOK 환경에서는 Amazon EC2에서 UEFI 보안 부팅할 때 사용할 수 없는 UEFI 펌웨어 키 등록의 일부 제한을 우회합니다. Amazon EC2에는 UEFI 변수 저장소의 키를 쉽게 조작할 수 있는 메커니즘이 있습니다.

키를 직접 등록하려면 기존 인스턴스 내에서 변수 저장소를 조작하거나([인스턴스 내에서 변수 저장소에 키 추가](#) 섹션 참조) 미리 채워진 바이너리 블록을 생성하면 됩니다([미리 채워진 변수 저장소가 있는 바이너리 블록 생성](#) 섹션 참조).

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.