



Red Hat Training and Certification

수강생 워크북 (ROLE)

Red Hat Enterprise Linux 9.0 RH199

RHCSA Rapid Track

엮음 6





Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



Network with tens of thousands of community members



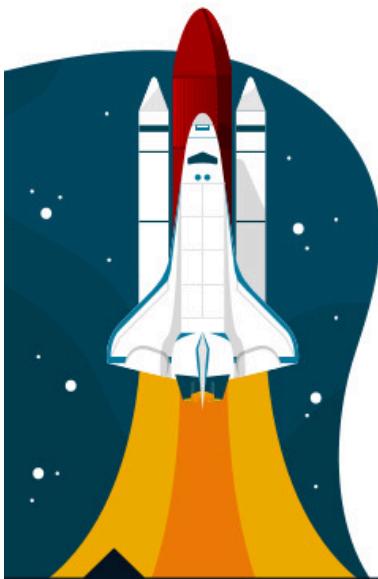
Engage in thousands of active conversations and posts



Join and interact with hundreds of certified training instructors



Unlock badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

Access free Red Hat training videos

Discover the latest Red Hat Training and Certification news

Connect with your instructor - and your classmates - before, after, and during your training course.

Join peers as you explore Red Hat products

Join the conversation learn.redhat.com



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

RHCSA Rapid Track

Red Hat Enterprise Linux 9.0 RH199
RHCSA Rapid Track
엮음 6 20230516
Publication date 20230516

Authors: Ashish Lingayat, Alejandra Ramírez Palacios, Antonio Marí Romero,
Bernardo Gargallo, Dallas Spohn, Ed Parenti, Jacob Pelchat,
Mike Kelly, Morgan Weetman, Patrick Gomez
Editor: Julian Cable

© 2023 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are © 2023 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to training@redhat.com or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is a trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

기여자: Adarsh Krishnan, David Sacco, Hemant Chauhan, Roberto Velazquez, Sajith Eyamkuzhy, Samik Sanyal, Yuvaraj Balaraju

문서 규칙	xi
	xi
소개	xiii
RHCSA Rapid Track	xiii
강의실 환경에 대한 오리엔테이션	xiv
랩 연습 수행	xviii
1. 시스템 액세스 및 지원 받기	1
쉘 프롬프트에서 텍스트 파일 편집	2
연습 가이드: 쉘 프롬프트에서 텍스트 파일 편집	6
SSH 키 기반 인증 구성	8
연습 가이드: SSH 키 기반 인증 구성	14
진단 보고서 생성	20
연습 가이드: 진단 보고서 생성	27
Red Hat Insights를 사용하여 문제 탐지 및 해결	29
퀴즈: Red Hat Insights를 사용하여 문제 탐지 및 해결	36
요약	38
2. 명령줄에서 파일 관리	39
Linux 파일 시스템 계층 구조 개념 설명	40
퀴즈: Linux 파일 시스템 계층 구조 개념 설명	42
파일 간 링크 만들기	46
연습 가이드: 파일 간 링크 만들기	50
쉘 확장을 사용하여 파일 이름 일치	52
퀴즈: 쉘 확장을 사용하여 파일 이름 일치	57
랩: 명령줄에서 파일 관리	61
요약	71
3. 로컬 사용자 및 그룹 관리	73
사용자 및 그룹 개념 설명	74
퀴즈: 사용자 및 그룹 개념 설명	77
수퍼유저 액세스 권한 얻기	81
연습 가이드: 수퍼유저 액세스 권한 얻기	86
로컬 사용자 계정 관리	91
연습 가이드: 로컬 사용자 계정 관리	94
로컬 그룹 계정 관리	97
연습 가이드: 로컬 그룹 계정 관리	100
사용자 암호 관리	103
연습 가이드: 사용자 암호 관리	107
랩: 로컬 사용자 및 그룹 관리	111
요약	117
4. 파일에 대한 액세스 제어	119
명령줄에서 파일 시스템 권한 관리	120
연습 가이드: 명령줄에서 파일 시스템 권한 관리	124
기본 권한 및 파일 액세스 관리	127
연습 가이드: 기본 권한 및 파일 액세스 관리	133
랩: 파일에 대한 액세스 제어	138
요약	144
5. SELinux 보안 관리	145
SELinux 적용 모드 변경	146
연습 가이드: SELinux 적용 모드 변경	151
SELinux 파일 컨텍스트 제어	154
연습 가이드: SELinux 파일 컨텍스트 제어	159
부울을 사용하여 SELinux 정책 조정	162
연습 가이드: 부울을 사용하여 SELinux 정책 조정	164

SELinux 문제 조사 및 해결	167
연습 가이드: SELinux 문제 조사 및 해결	171
랩: SELinux 보안 관리	175
요약	181
6. 시스템 성능 튜닝	183
프로세스 강제 종료	184
연습 가이드: 프로세스 강제 종료	189
프로세스 활동 모니터링	193
연습 가이드: 프로세스 활동 모니터링	197
튜닝 프로파일 조정	202
연습 가이드: 튜닝 프로파일 조정	209
프로세스 스케줄링에 미치는 영향	214
연습 가이드: 프로세스 스케줄링에 미치는 영향	218
랩: 시스템 성능 튜닝	222
요약	228
7. 향후 작업 예약	229
반복 실행 사용자 작업 예약	230
연습 가이드: 반복 실행 사용자 작업 예약	233
반복 실행 시스템 작업 예약	236
연습 가이드: 반복 실행 시스템 작업 예약	239
임시 파일 관리	242
연습 가이드: 임시 파일 관리	245
퀴즈: 향후 작업 예약	248
요약	250
8. 소프트웨어 패키지 설치 및 업데이트	251
Red Hat 지원을 위해 시스템 등록	252
퀴즈: Red Hat 지원을 위해 시스템 등록	256
DNF를 사용하여 소프트웨어 패키지 설치 및 업데이트	258
연습 가이드: DNF를 사용하여 소프트웨어 패키지 설치 및 업데이트	267
DNF 소프트웨어 리포지토리 활성화	272
연습 가이드: DNF 소프트웨어 리포지토리 활성화	275
랩: 소프트웨어 패키지 설치 및 업데이트	279
요약	285
9. 기본 스토리지 관리	287
파일 시스템 마운트 및 마운트 해제	288
연습 가이드: 파일 시스템 마운트 및 마운트 해제	291
파티션, 파일 시스템, 영구 마운트 추가	294
연습 가이드: 파티션, 파일 시스템, 영구 마운트 추가	303
스왑 공간 관리	307
연습 가이드: 스왑 공간 관리	311
랩: 기본 스토리지 관리	315
요약	323
10. 스토리지 스택 관리	325
논리 볼륨 생성 및 확장	326
연습 가이드: 논리 볼륨 생성 및 확장	336
랩: 스토리지 스택 관리	342
요약	349
11. 제어 서비스 및 부팅 프로세스	351
자동으로 시작된 시스템 프로세스 식별	352
연습 가이드: 자동으로 시작된 시스템 프로세스 식별	357
시스템 서비스 제어	361
연습 가이드: 시스템 서비스 제어	366

부팅 타겟 선택	370
연습 가이드: 부팅 타겟 선택	375
루트 암호 재설정	378
연습 가이드: 루트 암호 재설정	382
부팅 시 파일 시스템 문제 복구	384
연습 가이드: 부팅 시 파일 시스템 문제 복구	387
랩: 부팅 프로세스 제어	390
요약	396
12. 로그 분석 및 저장	397
시스템 로그 아키텍처 설명	398
퀴즈: 시스템 로그 아키텍처 설명	400
syslog 파일 검토	404
연습 가이드: syslog 파일 검토	408
시스템 저널 항목 검토	410
연습 가이드: 시스템 저널 항목 검토	415
시스템 저널 보존	418
연습 가이드: 시스템 저널 보존	421
정확한 시간 유지 관리	424
연습 가이드: 정확한 시간 유지 관리	428
랩: 로그 분석 및 저장	432
요약	437
13. 네트워킹 관리	439
네트워크 구성 유효성 검사	440
연습 가이드: 네트워크 구성 유효성 검사	446
명령줄에서 네트워킹 구성	449
연습 가이드: 명령줄에서 네트워킹 구성	456
네트워크 구성 파일 편집	461
연습 가이드: 네트워크 구성 파일 편집	465
호스트 이름 및 이름 확인 구성	469
연습 가이드: 호스트 이름 및 이름 확인 구성	473
랩: 네트워킹 관리	477
요약	482
14. 네트워크 연결 스토리지 액세스	483
NFS를 사용한 네트워크 연결 스토리지 관리	484
연습 가이드: NFS를 사용한 네트워크 연결 스토리지 관리	487
네트워크 연결 스토리지 자동 마운트	490
연습 가이드: 네트워크 연결 스토리지 자동 마운트	494
랩: 네트워크 연결 스토리지 액세스	499
요약	505
15. 네트워크 보안 관리	507
서버 방화벽 관리	508
연습 가이드: 서버 방화벽 관리	516
SELinux 포트 레이블 지정 제어	520
연습 가이드: SELinux 포트 레이블 지정 제어	523
랩: 네트워크 보안 관리	527
요약	535
16. 컨테이너 실행	537
컨테이너 개념	538
퀴즈: 컨테이너 개념	545
컨테이너 배포	547
연습 가이드: 컨테이너 배포	557
컨테이너 스토리지 및 네트워크 리소스 관리	563

연습 가이드: 컨테이너 스토리지 및 네트워크 리소스 관리	573
컨테이너를 시스템 서비스로 관리	579
연습 가이드: 컨테이너를 시스템 서비스로 관리	585
랩: 컨테이너 실행	591
요약	598

17. 종합 검토 599

종합 검토	600
랩: 부팅 문제 해결 및 서버 유지 관리	604
랩: 파일 시스템 및 스토리지 구성 및 관리	610
랩: 서버 보안 구성 및 관리	616
랩: 컨테이너 실행	626

문서 규칙

이 섹션에서는 모든 Red Hat 교육 과정에서 사용되는 다양한 규칙과 사례를 설명합니다.

권고

Red Hat 교육 과정에서는 다음과 같은 권고를 사용합니다.



참조

참조는 주제와 관련된 외부 설명서를 찾을 수 있는 위치를 설명합니다.



참고

"참고 사항"은 수행 중인 작업에 대한 팁, 바로 가기 또는 대체 접근법입니다. 참고 사항을 무시하더라도 큰 문제가 발생하지는 않지만 더 쉬운 메서드를 알려주는 것을 놓칠 수도 있습니다.



중요

이 권고에서는 현재 세션에만 적용되는 구성 변경 사항 또는 업데이트를 적용하기 전에 다시 시작해야 하는 서비스 등 놓치기 쉬운 내용을 자세히 정의합니다. 이 권고를 무시하는 경우 데이터 손실이 발생하지는 않지만 불편하고 당황스러운 경우가 발생할 수 있습니다.



경고

경고는 무시하지 말아야 합니다. 이 권고를 무시할 경우 데이터 손실이 발생할 수 있습니다.

포함 언어

Red Hat 교육에서는 현재 잠재적으로 불쾌한 용어를 제거하기 위해 다양한 영역에서 언어의 사용을 검토 중입니다. 이 프로세스는 계속 진행되며 Red Hat 교육 과정에서 다른 제품과 서비스에 맞게 조정해야 합니다. Red Hat에서는 이 프로세스를 기다려 주신 데 대해 감사드립니다.

소개

RHCSA Rapid Track

Red Hat Certified System Administrator Rapid Track(RH199) 교육 과정은 Linux를 많이 사용하는 IT 전문가를 대상으로 Red Hat Enterprise Linux 시스템 관리에 대한 신속한 교육 과정으로 설계되었습니다. Linux 명령줄에 대한 기초 지식을 갖춘 수강생은 단일 Red Hat Enterprise Linux 시스템을 관리하기 위한 주요 작업을 학습합니다. 이 과정은 신속 과정입니다. 강사 주도 교육 과정으로 제공되는 이 교육 과정은 일반적으로 Red Hat System Administration I 및 Red Hat System Administration II에서 10일 동안 다루는 교육 콘텐츠를 5일 동안 다룹니다. 이 교육 과정에서는 다른 두 교육 과정의 일부 개념을 간략하게만 다루거나 다루지 않습니다.

교육 과정 목표

- Red Hat System Administration I(RH124) 교육 과정에서 얻은 기술을 확장합니다.
- RHCSA 인증 Red Hat Enterprise Linux 시스템 관리자에게 필요한 기술을 확보합니다.

대상

- RHCSA Rapid Track(RH199)은 Linux를 많이 사용하는 IT 전문가를 대상으로 Red Hat Enterprise Linux 시스템 관리에 대한 신속한 교육 과정으로 설계되었습니다. 수강생은 쉘 프롬프트에서 일반적인 Linux 명령을 실행하는 데 익숙해야 합니다. 이러한 지식이 부족한 수강생은 대신 Red Hat System Administration I(RH124)을 수강하는 것이 좋습니다.

사전 요구 사항

다음 활동 목록에 익숙해질 것으로 예상됩니다.

- 상대 및 절대 경로를 사용하여 파일 및 디렉터리를 만듭니다.
- 파일을 복사하고 이동합니다.
- 아카이브 파일을 생성하고 복원합니다.
- 파일 시스템 권한을 해석합니다.
- 시스템에서 파일을 찾습니다.
- 간단한 Bash 스크립트를 작성합니다.

강의실 환경에 대한 오리엔테이션

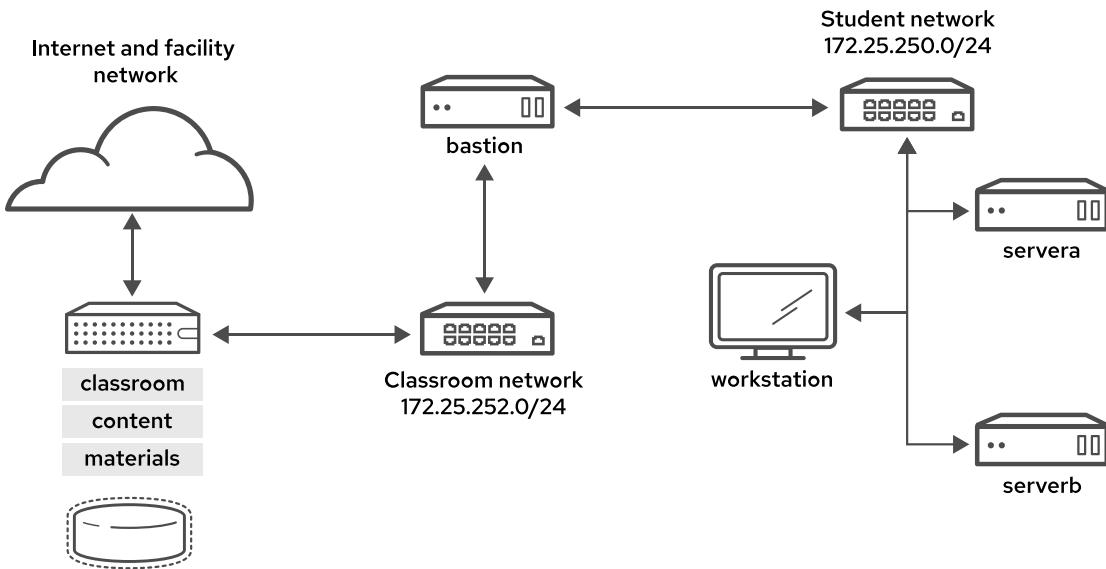


그림 0.1: 강의실 환경

이 교육 과정에서 실습 활동에 사용되는 주요 컴퓨터 시스템은 **workstation**입니다. 또한 수강생은 **servera** 및 **serverb** 활동을 위해 다른 두 개의 시스템을 사용합니다. 3개 시스템은 모두 `lab.example.com` DNS 도메인에 있습니다.

모든 수강생 컴퓨터 시스템의 표준 사용자 계정은 **student**이고 암호는 **student**입니다. 모든 수강생 시스템의 **root** 암호는 **redhat**입니다.

강의실 시스템

시스템 이름	IP 주소	역할
<code>bastion.lab.example.com</code>	172.25.250.254	학생 개인 네트워크를 교실 서버에 연결하는 게이트웨이 시스템 (항상 실행되어야 함)
<code>workstation.lab.example.com</code>	172.25.250.9	시스템 관리를 위한 그래픽 워크스테이션
<code>servera.lab.example.com</code>	172.25.250.10	관리 서버 "A"
<code>serverb.lab.example.com</code>	172.25.250.11	관리 서버 "B"

bastion의 주요 기능은 수강생 시스템을 연결하는 네트워크와 강의실 네트워크 간의 라우터 역할을 하는 것입니다. **bastion**이 다운되면 다른 수강생 시스템은 개별 수강생 네트워크에 있는 시스템에만 액세스할 수 있습니다.

강의실 안의 여러 시스템은 지원 서비스를 제공합니다. 두 개의 서버 `content.example.com` 및 `materials.example.com`에는 실습 활동에 사용되는 소프트웨어와 랩 자료가 있습니다. 이러한 서버를

소개

사용하는 방법에 대한 정보는 활동 지침에 제공됩니다. **workstation** 가상 시스템에서 해당 활동을 제공합니다. 랩 환경을 올바르게 사용하려면 **classroom** 및 **bastion** 모두 항상 실행되고 있어야 합니다.



참고

servera 또는 **serverb**에 로그인하는 경우 **cockpit** 활성화에 관한 메시지가 표시될 수 있습니다. 이 메시지는 무시해도 됩니다.

```
[student@workstation ~]$ ssh student@serverb
Warning: Permanently added 'serverb,172.25.250.11' (ECDSA) to the list of
known hosts.
Activate the web console with: systemctl enable --now cockpit.socket

[student@serverb ~]$
```

가상 시스템 제어

ROLE(Red Hat 온라인 학습) 강의실의 원격 컴퓨터가 할당됩니다. 자기 주도식 교육 과정에 액세스할 때 사용하는 웹 애플리케이션의 호스트 주소는 rol.redhat.com [<http://rol.redhat.com>]입니다. Red Hat Customer Portal 사용자 자격 증명을 사용하여 이 사이트에 로그인합니다.

가상 시스템 제어

강의실 환경의 가상 컴퓨터는 웹 페이지 인터페이스 컨트롤을 통해 제어합니다. 각 강의실 가상 시스템의 상태는 **Lab Environment** 탭에 표시됩니다.

VM Name	Status	Action	Open Console
bastion	active	ACTION -	OPEN CONSOLE
classroom	active	ACTION -	OPEN CONSOLE
servera	building	ACTION -	OPEN CONSOLE
serverb	building	ACTION -	OPEN CONSOLE
workstation	active	ACTION -	OPEN CONSOLE

그림 0.2: 교육 과정 랩 환경 관리 페이지의 예

시스템 상태

가상 시스템 상태	설명
building	가상 시스템을 생성 중입니다.

가상 시스템 상태	설명
active	가상 시스템이 실행 중이고 사용 가능합니다. 방금 시작된 가상 시스템에서도 서비스를 시작할 수 있습니다.
stopped(중지됨)	가상 시스템이 완전히 종료되었습니다. 가상 시스템을 시작하면 종료되기 전과 동일한 상태로 부팅됩니다. 디스크 상태가 유지됩니다.

강의실 작업

버튼 또는 작업	설명
CREATE	ROLE 강의실을 생성합니다. 이 강의실에 필요한 모든 가상 시스템을 생성하고 시작합니다. 생성하는 데 몇 분이 소요될 수 있습니다.
CREATING	ROLE 강의실 가상 시스템을 생성 중입니다. 이 강의실에 필요한 모든 가상 시스템을 생성하고 시작합니다. 생성하는 데 몇 분이 소요될 수 있습니다.
DELETE	ROLE 강의실을 삭제합니다. 강의실의 모든 가상 컴퓨터를 제거합니다. 시스템의 디스크에 저장된 모든 작업이 손실됩니다.
START	강의실의 모든 가상 시스템을 시작합니다.
STARTING	강의실의 모든 가상 시스템을 시작 중입니다.
STOP	강의실의 모든 가상 시스템을 중지합니다.

시스템 작업

버튼 또는 작업	설명
OPEN CONSOLE	새 브라우저 탭에서 가상 시스템의 시스템 콘솔에 연결합니다. 필요에 따라 가상 시스템에 직접 로그인하고 명령을 실행할 수 있습니다. 보통은 workstation 가상 시스템에만 로그인한 다음 ssh를 사용하여 다른 가상 시스템에 연결합니다.
ACTION > Start	가상 시스템을 시작합니다(전원 켜기).
ACTION > Shutdown	가상 시스템을 정상적으로 종료하고 디스크 콘텐츠를 유지합니다.
ACTION > Power Off	가상 시스템을 강제로 종료해도 디스크 콘텐츠는 유지됩니다. 이 경우 실제 시스템에서 전원을 분리하는 것과 동일합니다.
ACTION > Reset	가상 시스템을 강제로 종료하고 관련 스토리지를 초기 상태로 재설정합니다. 시스템 디스크에 저장된 모든 작업이 손실됩니다.

연습을 시작할 때 단일 가상 시스템 노드를 재설정하라는 지침이 나타나면 특정 가상 시스템에 대해서만 ACTION > Reset을 클릭합니다.

연습을 시작할 때 가상 시스템을 모두 재설정하라는 지침이 나타나면 목록의 모든 가상 시스템에서 ACTION > Reset을 클릭합니다.

교육 과정 시작 시 강의실 환경을 원래 상태로 되돌리려면 DELETE를 클릭하여 전체 강의실 환경을 제거할 수 있습니다. 랩이 삭제된 후 CREATE를 클릭하여 새 강의실 시스템을 프로비전합니다.



경고

DELETE 작업은 실행 취소할 수 없습니다. 강의실 환경에서 완료한 모든 작업이 손실됩니다.

자동 중지 및 자동 삭제 타이머

Red Hat 온라인 학습에 등록한 경우 일정 시간 동안 컴퓨터를 사용할 수 있습니다. 할당된 시간을 절약하기 위해 ROLE 강의실에서는 타이머를 사용하며, 해당 타이머가 만료되면 강의실 환경을 종료하거나 삭제합니다.

타이머를 조정하려면 교육 과정 관리 페이지의 맨 아래에 있는 두 개의 **+** 버튼을 찾으십시오. 자동 중지 **+** 버튼을 클릭하여 자동 중지 타이머에 시간을 더 추가합니다. 자동 삭제 **+** 버튼을 클릭하여 자동 삭제 타이머에 일수를 추가합니다. 최대 자동 중지 시간은 11시간이고 최대 자동 삭제 기간은 14일입니다. 환경이 예기치 않게 종료되지 않도록, 작업하는 동안 타이머 설정에 유지하십시오. 타이머를 너무 길게 설정하여 할당된 서브스크립션 시간을 낭비하지 않도록 주의하십시오.

랩 연습 수행

이 교육 과정에는 다음과 같은 랩 활동 유형이 표시될 수 있습니다.

- 안내에 따른 연습은 프레젠테이션 섹션을 따르는 실습 연습입니다. 수행할 절차를 단계별로 안내합니다.
- 퀴즈는 일반적으로 지식 기반 학습을 확인할 때 또는 다른 이유로 인해 실습 활동이 비실용적일 때 사용됩니다.
- 장 종료 랩은 학습을 확인하는 데 도움이 되며 채점 가능한 실습 활동입니다. 해당 장의 안내에 따른 연습에 따라 일련의 고급 단계를 수행하지만, 단계가 모든 명령을 안내하지는 않습니다. 솔루션은 단계별 연습과 함께 제공됩니다.
- 교육 과정이 끝나면 종합 검토 랩이 사용됩니다. 종합 검토 랩은 채점 가능한 실습 활동이기도 하며, 전체 교육 과정의 내용을 다룰 수 있습니다. 특정 단계를 수행하지 않고 활동에서 수행할 작업을 지정합니다. 또한 솔루션은 사양을 충족하는 단계별 연습과 함께 제공됩니다.

각 실습 활동을 시작할 때 랩 환경을 준비하려면 활동 지침에서 지정된 활동 이름을 사용하여 **lab start** 명령을 실행합니다. 마찬가지로 각 실습 활동이 끝나면 동일한 활동 이름으로 **lab finish** 명령을 실행하여 활동 후에 정리합니다. 교육 과정의 각 실습 활동은 고유한 이름이 있습니다.

연습 스크립트를 실행하는 구문은 다음과 같습니다.

```
[student@workstation ~]$ lab action exercise
```

작업으로는 **start**, **grade** 또는 **finish**를 선택할 수 있습니다. 모든 연습에서 **start**과 **finish**를 사용할 수 있습니다. 장 종료 랩과 종합 검토 랩에서만 **grade**를 지원합니다.

시작

start 작업은 연습을 시작하는 데 필요한 리소스를 확인합니다. 여기에는 설정 구성, 리소스 생성, 사전 요구 사항 서비스 확인, 이전 연습에서 필요한 결과 확인 등이 포함될 수 있습니다. 선행 연습을 진행하지 않은 경우에도 언제든지 연습을 진행할 수 있습니다.

grade

채점 가능한 활동의 경우 **grade** 작업은 작업을 평가하도록 **lab** 명령을 지시하고, 각각에 대해 **PASS** 또는 **FAIL** 상태로 채점 기준 목록을 표시합니다. 모든 기준에 대해 **PASS** 상태를 달성하려면 실패를 수정하고 **grade** 작업을 다시 실행하십시오.

finish

finish 작업은 연습 중에 구성된 리소스를 정리합니다. 연습은 원하는 만큼 진행할 수 있습니다.

lab 명령은 탭 완료를 지원합니다. 예를 들어 시작할 수 있는 모든 연습을 나열하려면 **lab start**를 입력한 다음 **Tab** 키를 두 번 누릅니다.

1장

시스템 액세스 및 지원 받기

목적

텍스트 파일을 편집하고, 로컬 및 원격 Linux 시스템에 로그인하며, Red Hat Support 및 Red Hat Insights를 통해 제공되는 문제 해결 방법을 조사합니다.

목표

- vim 편집기를 사용하여 명령 줄에서 텍스트 파일을 생성하고 편집합니다.
- 암호 없이 원격 시스템에 안전하게 로그인하기 위해 키 기반 인증을 사용하도록 사용자 계정을 구성합니다.
- Red Hat Customer Portal 주요 리소스를 설명하고 사용하여 Red Hat 설명서 및 Knowledgebase에서 정보를 찾습니다.
- Red Hat Insights를 사용하여 서버의 문제를 분석하고 문제를 수정하거나 해결한 다음, 솔루션이 성공했는지 확인합니다.

섹션

- 텘 프롬프트에서 텍스트 파일 편집(안내에 따른 연습)
- SSH 키 기반 인증 구성(안내에 따른 연습)
- 진단 보고서 생성(안내에 따른 연습)
- Red Hat Insights를 사용하여 문제 탐지 및 해결(퀴즈)

쉘 프롬프트에서 텍스트 파일 편집

목표

vim 편집기를 사용하여 명령 줄에서 텍스트 파일을 생성하고 편집합니다.

Vim을 사용하여 파일 편집

Linux의 기본 설계 원칙은 정보 및 구성 설정을 텍스트 기반 파일에 저장하도록 지원한다는 것입니다. 이러한 파일은 설정 목록, INI 유사 형식, 구조적 XML 또는 YAML 등의 다양한 구조를 따릅니다. 텍스트 기반 구조에 파일을 저장할 경우 텍스트 편집기로 편집할 수 있다는 이점이 있습니다.

Vim은 Linux 및 UNIX 시스템에 배포된 **vi** 편집기의 개선된 버전입니다. Vim은 고도로 구성 가능한 효율적인 편집기로 분할 화면 편집, 컬러 포맷팅, 텍스트 편집을 위한 강조 표시 기능 등을 제공합니다.

Vim 편집기의 이점

시스템에서 텍스트 전용 쉘 프롬프트를 사용하는 경우 파일을 편집하는데 하나 이상의 텍스트 편집기를 사용하는 방법을 알아야 합니다. 그런 다음, 터미널 창에서 또는 **ssh** 명령이나 웹 콘솔을 통해 원격 로그인에서 텍스트 기반 구성 파일을 편집할 수 있습니다. 서버의 파일을 편집하기 위해 그래픽 데스크탑에 액세스할 필요가 없으며 해당 서버에서 그래픽 데스크탑 환경을 실행할 필요가 없을 수도 있습니다.

Vim을 배워야 하는 주요 이유는 텍스트 기반 파일을 편집하기 위해 서버에 거의 항상 기본적으로 설치되기 때문입니다. Portable Operating System Interface, 즉 POSIX 표준은 Linux에서 **vi** 편집기를 지정했으며, 다른 많은 UNIX 유사 운영 체제도 마찬가지입니다.

Vim이 다른 표준 운영 체제 또는 배포판에서 **vi** 구현으로 사용되는 경우도 많습니다. 예를 들어, macOS는 현재 기본적으로 가벼운 버전의 Vim 설치를 포함합니다. 따라서 Linux에 대해 배운 Vim 기술은 다른 곳에서도 유용할 수 있습니다.

Vim 사용 시작

두 패키지 중 하나를 사용하여 Red Hat Enterprise Linux의 Vim 편집기를 설치할 수 있습니다. 두 패키지는 텍스트 기반 파일을 편집하기 위한 다양한 기능과 Vim 명령을 제공합니다.

vim-minimal 패키지를 사용하면 핵심 기능이 있는 **vi** 편집기를 설치할 수 있습니다. 이 경량 설치에는 핵심 기능과 기본 **vi** 명령만 포함됩니다. **vi** 명령을 사용하여 편집할 파일을 열 수 있습니다.

```
[user@host ~]$ vi filename
```

또는 **vim-enhanced** 패키지를 사용하여 Vim 편집기를 설치할 수 있습니다. 이 패키지는 훨씬 더 포괄적인 기능 세트, 온라인 도움말 시스템 및 튜토리얼 프로그램을 제공합니다. 이 향상된 모드로 Vim을 시작하려면 **vim** 명령을 사용합니다.

```
[user@host ~]$ vim filename
```

Vim 편집기의 핵심 기능은 두 명령에서 모두 사용할 수 있습니다.

vim-enhanced 가 설치된 경우 헬 별칭이 설정되어 있으므로 일반 사용자가 **vi** 명령을 실행하면 자동으로 **vim** 명령을 대신 가져옵니다. 이 별칭은 **root** 사용자 및 시스템 서비스에서 사용하고 UID가 200 미만인 다른 사용자에게는 적용되지 않습니다.

vim-enhanced 가 설치되어 있고 일반 사용자가 **vi** 명령을 사용하려는 경우 **\vi** 명령을 사용하여 별칭을 일시적으로 재정의해야 할 수 있습니다. **\vi --version** 및 **vim --version**을 사용하여 두 명령의 기능 세트를 비교할 수 있습니다.

Vim 운영 모드

Vim 편집기는 명령 모드, 확장 명령 모드, 편집 모드, 시각적 모드와 같은 다양한 작업 모드를 제공합니다. 키 입력의 효과는 모드마다 다르므로 Vim 사용자는 항상 현재 모드를 확인합니다.

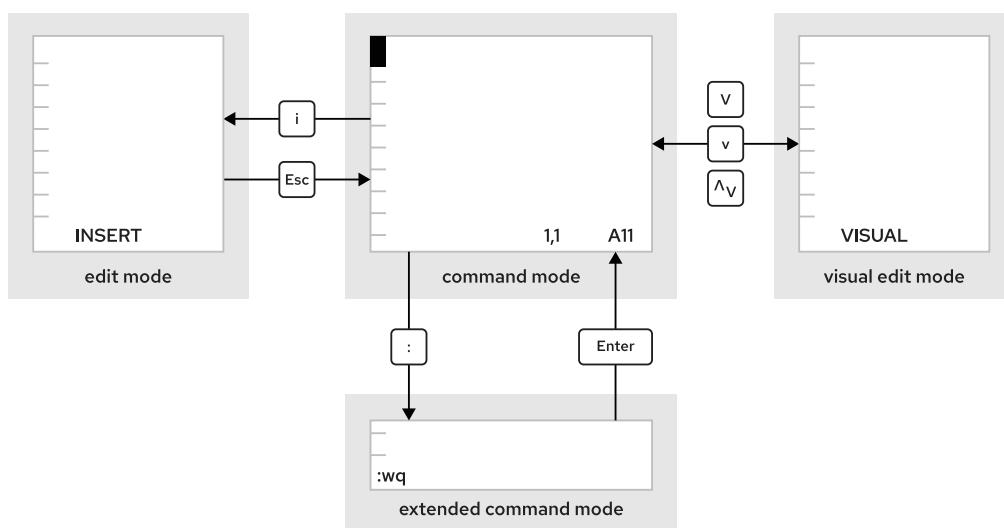


그림 1.1: Vim 모드 전환

Vim을 처음 열면 네비게이션, 잘라내어 붙여넣기, 기타 텍스트 수정에 사용되는 명령 모드로 시작됩니다. 필요한 키 입력을 눌러 특정 편집 기능에 액세스합니다.

- **i** 키를 누르면 입력한 모든 텍스트가 파일 내용이 되는 삽입 모드가 시작됩니다. **Esc**를 누르면 명령 모드로 돌아갑니다.
- **v** 키를 누르면 텍스트 조작을 위해 여러 문자를 선택할 수 있는 시각적 모드가 시작됩니다. 여러 줄인 경우 **Shift+V**, 블록 선택인 경우 **Ctrl+V**를 사용합니다. 시각적 모드를 종료하려면 **v**, **Shift+V** 또는 **Ctrl+V** 키 입력을 사용합니다.
- **:** 키를 누르면 파일을 작성하여 저장하기, Vim 편집기 종료 등의 작업이 가능한 확장 명령 모드가 시작됩니다.



참고

Vim에서 사용 중인 모드가 확실하지 않은 경우 **Esc** 키를 몇 번 눌러 명령 모드로 돌아갑니다. 명령 모드에서는 **Esc** 키를 반복해서 눌러도 안전합니다.

최소한의 기본 Vim 워크플로

Vim에서는 고급 편집 작업을 위한 효율적이고 체계적인 키 입력이 가능합니다. 연습을 통해 숙련될 수 있지만 새로운 사용자에게는 Vim의 기능이 어렵게 여겨질 수도 있습니다.

다음 Vim 키와 명령에 대해 알아보는 것이 좋습니다.

- **u** 키를 누르면 가장 최근 편집 내용이 실행 취소됩니다.
- **x** 키를 누르면 단일 문자가 삭제됩니다.
- **:w** 명령은 파일을 작성하여 저장하고, 추가 편집을 위해 계속 명령 모드를 유지합니다.
- **:wq** 명령은 파일을 작성하여 저장하고 Vim을 종료합니다.
- **:q!** 명령은 Vim을 종료하고 마지막 작성 이후 변경된 파일 내용을 모두 취소합니다.

이러한 명령을 배우면 Vim 사용자가 편집 작업을 수행하는 데 도움이 됩니다.

기존 텍스트 재정렬

Vim에서 **y** 및 **p** 명령 문자를 사용하여 yank 및 put(복사하여 붙여넣기)을 수행할 수 있습니다. 먼저 선택할 첫 번째 문자에 커서를 놓고 시각적 모드를 시작합니다. 화살표 키를 사용하여 시각적 선택 영역을 확장합니다. 준비되었으면 **y**를 눌러 선택한 내용을 메모리로 yank(복사)합니다. 커서를 새 위치에 놓은 다음, **p**를 눌러 선택한 내용을 커서 위치에 붙여넣습니다.

Vim의 시각적 모드

시각적 모드는 여러 줄과 열의 텍스트를 강조 표시하고 조작하는데 유용합니다. 다음 키 조합을 사용하여 Vim에서 다양한 시각적 모드를 시작할 수 있습니다.

- 문자 모드: **v**
- 줄 모드: **Shift+v**
- 블록 모드: **Ctrl+v**

문자 모드는 텍스트 블록의 문장을 강조 표시합니다. **VISUAL**이라는 단어가 화면 맨 아래에 표시됩니다. **v**를 누르면 시각적 문자 모드가 시작됩니다. **Shift+v**를 누르면 줄 모드가 시작됩니다. **VISUAL LINE**이 화면 맨 아래에 표시됩니다.

시각적 블록 모드는 데이터 파일 조작에 이상적입니다. **Ctrl+v** 키를 누르면 커서에서 시각적 블록이 시작됩니다. **VISUAL BLOCK**이 화면 맨 아래에 표시됩니다. 화살표 키를 사용하여 변경할 섹션을 강조 표시합니다.



참고

먼저 기본 Vim 기능에 익숙해지는 시간을 가져보십시오. 그런 다음 Vim 키 입력을 더 학습하여 Vim 어휘를 확장합니다.

이 섹션의 연습에서는 **vimtutor** 명령을 사용합니다. **vim-enhanced** 패키지 튜토리얼은 Vim의 핵심 기능을 배우는 훌륭한 방법입니다.

Vim 구성 파일

/etc/vimrc 및 **~/.vimrc** 구성 파일은 전체 시스템 또는 특정 사용자에 대한 **vim** 편집기의 동작을 각각 변경합니다. 이러한 구성 파일 내에서 기본 탭 간격, 구문 강조 표시, 색 구성표 등의 동작을 지정할 수 있습니다. **vim** 편집기의 동작을 수정하면 구문 요구 사항이 엄격한 YAML과 같은 언어로 작업할 때 특히 유용합니다. YAML 파일을 편집하는 동안 기본 탭 정지(**ts** 문자로 표시됨)를 두 개의 공백으로 설정하는 다음 **~/.vimrc** 파일을 살펴보십시오. 이 파일에는 모든 파일을 편집하는 동안 줄 번호를 표시하는 **set number** 매개 변수도 포함되어 있습니다.

```
[user@host ~]$ cat ~/.vimrc
autocmd FileType yaml setlocal ts=2
set number
```

vimrc 구성 옵션의 전체 목록은 참조에서 확인할 수 있습니다.



참조

vim(1) 도움말 페이지

:help에 있는 **vim** 명령(**vim-enhanced** 패키지가 설치된 경우).

Vim 참조 설명서: Vim 옵션

<https://vimhelp.org/options.txt.html#options.txt>

▶ 연습 가이드

쉘 프롬프트에서 텍스트 파일 편집

이 연습에서는 **vimtutor** 명령을 사용하여 **vim** 편집기에서 기본 편집 기술을 연습합니다.

결과

- Vim을 사용하여 파일을 편집합니다.
- **vimtutor** 명령을 사용하여 Vim을 능숙하게 익힙니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start edit-editfile
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **vimtutor** 명령을 실행합니다. 인사말 화면을 읽고 1.1 단원을 수행합니다.

프레젠테이션에서 키보드 화살표 키는 창을 탐색하는 데 도움이 됩니다. 처음에 **vi** 편집기가 개발되었을 때는 사용자가 화살표 키나 화살표 키에 대한 키보드 매핑을 사용하여 커서를 움직일 수 없었습니다. 따라서 **vi** 편집기는 편리하게 그룹화된 **h**, **j**, **k**, **l** 등의 표준 문자 키로 명령을 사용하여 커서를 움직이도록 원래 설계되었습니다.

다음은 키를 기억하는 한 가지 방법입니다.

hang **back**, jump **down**, kick **up**, leap **forward**.

```
[student@servera ~]$ vimtutor
```

- ▶ 3. **vimtutor** 창에서 단원 1.2를 수행합니다.

이 단원에서는 원치 않는 변경 사항을 유지하지 않고 종료하는 방법을 설명합니다. 모든 변경 사항이 손실됩니다. 중요한 파일을 잘못된 상태로 두는 것보다 변경 사항을 잊는 것이 더 바람직한 경우도 있습니다.

- ▶ 4. **vimtutor** 창에서 단원 1.3을 수행합니다.

Vim에서는 빠르고 효율적인 키 입력을 통해 정확한 개수의 단어, 줄, 문장 또는 단락을 삭제할 수 있습니다. 단일 문자를 삭제하는 데 **x** 키를 사용하여 편집할 수 있습니다.

▶ 5. **vimtutor** 창에서 단원 1.4을 수행합니다.

대부분의 편집 작업에서 처음 누르는 키는 **i** (입력) 키입니다.

▶ 6. **vimtutor** 창에서 단원 1.5을 수행합니다.

이전 강의에서는 편집 모드를 시작하기 위한 **i** 명령에 대해서만 설명했습니다. 이 단원에서는 삽입 모드에서 커서 배치를 변경하는데 사용할 수 있는 다른 키 입력을 보여줍니다. 삽입 모드에서는 입력한 모든 텍스트가 파일 내용을 변경합니다.

▶ 7. **vimtutor** 창에서 단원 1.6을 수행합니다.

:wq를 입력하여 파일을 저장하고 편집기를 종료합니다.

▶ 8. **vimtutor** 창에서 단원 1 요약을 읽습니다.

vimtutor 명령에는 6개의 다단계 단원이 더 포함되어 있습니다. 이러한 단원은 이 교육 과정의 일부로 할당되지 않았지만, 언제든지 편하게 살펴보고 자세히 알아볼 수 있습니다.

▶ 9. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish edit-editfile
```

이것으로 섹션을 완료합니다.

SSH 키 기반 인증 구성

목표

암호 없이 원격 시스템에 안전하게 로그인하기 위해 키 기반 인증을 사용하도록 사용자 계정을 구성합니다.

SSH 키 기반 인증

PKI(공개 키 암호화)를 기반으로 하는 키 기반 인증을 활성화한 SSH 서버에 암호 없이 액세스하도록 계정을 구성할 수 있습니다.

계정을 준비하려면 암호화 관련 키 파일 쌍을 생성합니다. 하나의 키는 개인용이며 사용자만 보유합니다. 두 번째 키는 비밀이 아닌 관련 공개 키입니다. 개인 키는 인증 자격 증명 역할을 하며 안전하게 저장해야 합니다. 공개 키는 원격으로 액세스할 서버의 계정에 복사되어 개인 키 사용을 확인합니다.

원격 서버에서 계정에 로그인하면 원격 서버에서 공개 키를 사용하여 과제 메시지를 암호화하여 SSH 클라이언트로 보냅니다. 그런 다음 SSH 클라이언트는 이 메시지를 해독할 수 있음을 증명해야 합니다. 이는 연결된 개인 키가 있음을 보여줍니다. 확인에 성공하면 요청이 신뢰받고 암호를 제공하지 않고도 액세스할 수 있습니다.

암호는 쉽게 습득하거나 도난당할 수 있지만 안전하게 저장된 개인 키는 손상하기가 더 어렵습니다.

SSH 키 생성

`ssh-keygen` 명령을 사용하여 키 쌍을 생성합니다. 기본적으로 `ssh-keygen` 명령은 개인 키와 공개 키를 `~/.ssh/id_rsa` 및 `~/.ssh/id_rsa.pub` 파일에 저장하지만 다른 이름을 지정할 수 있습니다.

```
[user@host ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): Enter
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:vxutUNPi03QDCyvkYm1 user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]---+
|           |
|           |
|   o   o   |
| . = o   o . |
| o + = S E . |
| ..o o + * + |
| .+% o . + B . |
| =*o0 . . + * |
| ++. . +. |
+---[SHA256]---
```

개인 키를 암호화하는 데 사용되는 암호를 **ssh-keygen**에 제공할 수 있습니다. 액세스하려는 사람이 개인 키를 사용할 수 없도록 암호를 사용하는 것이 좋습니다. 암호를 설정한 경우 개인 키를 사용할 때마다 암호를 입력해야 합니다. 사용하기 위해 네트워크를 통해 일반 텍스트로 보내야 하는 암호(password)와 달리, 암호(passphrase)는 개인 키를 사용하기 전에 암호를 해독하기 위해 로컬에서 사용됩니다.

로그인 세션에서 처음 사용할 때 암호를 캐시한 다음, 동일한 로그인 세션에서 이후에 개인 키를 사용할 때마다 암호를 제공하는 **ssh-agent** 키 관리자를 로컬에서 사용할 수 있습니다. **ssh-agent** 명령에 대해서는 이 섹션의 뒷부분에서 설명합니다.

다음 예제에서는 공개 키를 사용하여 암호로 보호된 개인 키를 생성합니다.

```
[user@host ~]$ ssh-keygen -f .ssh/key-with-pass
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): your_passphrase
Enter same passphrase again: your_passphrase
Your identification has been saved in .ssh/key-with-pass.
Your public key has been saved in .ssh/key-with-pass.pub.
The key fingerprint is:
SHA256:w3GGB7EyHUri4a0cNPKmhNKS7dl1YsMVLvFZJ77VxAo user@host.lab.example.com
The key's randomart image is:
+---[RSA 2048]----+
|       . + = . o ...
|       = B XEo o .
|       . o O X = . . .
|       = = = B = o .
|= + * * S .
|.+ = o + .
| + .
|
|
+---[SHA256]-----+
```

ssh-keygen 명령의 **-f** 옵션은 키를 저장할 파일을 지정합니다. 앞의 예제에서 **ssh-keygen** 명령은 `/home/user/.ssh/key-with-pass` 및 `/home/user/.ssh/key-with-pass.pub` 파일에 키 쌍을 저장했습니다.



경고

새 **ssh-keygen** 명령을 사용하는 동안 기본 `id_rsa` 쌍을 포함하여 기존 키 쌍의 이름을 지정하면 기존 키 쌍을 덮어쓰게 되며, 해당 파일의 백업이 있는 경우에만 복원할 수 있습니다. 키 쌍을 덮어쓰면 원격 서버에서 해당 공용 키로 구성한 계정에 액세스하는 데 필요한 원래 개인 키가 손실됩니다.

로컬 개인 키를 복원할 수 없는 경우 각 서버에서 이전 공개 키를 교체할 새 공개 키를 배포할 때까지 원격 서버에 액세스할 수 없습니다. 키를 덮어쓰거나 분실할 때를 대비하여 항상 키 백업을 생성합니다.

생성된 SSH 키는 기본적으로 홈 디렉터리의 `.ssh` 하위 디렉터리에 저장됩니다. 정확하게 작동하려면 개인 키가 속한 사용자만 개인 키를 읽고 쓸 수 있어야 합니다(8진수 권한 600). 공개 키는 민감하지 않으며 시스템의 모든 사용자가 읽을 수도 있습니다(8진수 권한 644).

공개 키 공유

액세스를 위해 원격 계정을 구성하려면 공개 키를 원격 시스템에 복사합니다. **ssh-copy-id** 명령은 SSH 키 쌍의 공개 키를 원격 시스템에 복사합니다. **ssh-copy-id** 명령을 사용하여 특정 공개 키를 지정하거나 기본 `~/.ssh/id_rsa.pub` 파일을 사용할 수 있습니다.

```
[user@host ~]$ ssh-copy-id -i .ssh/key-with-pass.pub user@remotehost
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/user/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
user@remotehost's password: redhat
Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'user@remotehost'"
and check to make sure that only the key(s) you wanted were added.
```

공개 키를 배치한 후 해당 개인 키를 사용하여 원격 액세스를 테스트합니다. 구성이 올바르면 계정 암호를 묻는 메시지가 표시되지 않고 원격 시스템의 계정에 액세스합니다. 개인 키 파일을 지정하지 않으면 **ssh** 명령은 기본 `~/.ssh/id_rsa` 파일을 사용합니다(있는 경우).



중요

개인 키를 보호하기 위해 암호를 구성한 경우 처음 사용할 때 SSH에서 암호를 요청합니다. 그러나 키 인증에 성공하면 계정 암호를 묻는 메시지가 표시되지 않습니다.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost
Enter passphrase for key '.ssh/key-with-pass': your_passphrase
...output omitted...
[user@remotehost ~]$
```

키 관리자를 사용하는 비대화형 인증

암호를 사용하여 개인 키를 암호화한 경우 인증을 위해 개인 키를 사용할 때마다 암호를 입력해야 합니다. 그러나 암호를 캐시하도록 **ssh-agent** 키 관리자를 구성할 수 있습니다. 그러면 SSH를 사용할 때마다 **ssh-agent** 키 관리자가 암호를 제공합니다. 키 관리자를 사용하면 편리할 뿐만 아니라 다른 사람이 암호를 관찰할 수 있는 기회가 줄어 보안이 향상될 수 있습니다.

로그인할 때 자동으로 시작되도록 **ssh-agent** 키 관리자를 구성할 수 있습니다. GNOME 그래픽 데스크탑 환경은 **ssh-agent** 키 관리자를 자동으로 시작하고 구성할 수 있습니다. 텍스트 환경에 로그인하는 경우 각 세션에서 **ssh-agent** 프로그램을 수동으로 시작해야 합니다. 다음 명령을 사용하여 **ssh-agent** 프로그램을 시작합니다.

```
[user@host ~]$ eval $(ssh-agent)
Agent pid 10155
```

ssh-agent 명령을 수동으로 시작하면 추가 쉘 명령을 실행하여 **ssh-add** 명령과 함께 사용하는데 필요한 환경 변수를 설정합니다. **ssh-add** 명령을 사용하여 키 관리자에 개인 키 암호를 수동으로 로드할 수 있습니다.

다음 예제 `ssh-add` 명령은 기본 `~/.ssh/id_rsa` 파일과 `~/.ssh/key-with-pass` 파일에서 개인 키를 차례로 추가합니다.

```
[user@host ~]$ ssh-add  
Identity added: /home/user/.ssh/id_rsa (user@host.lab.example.com)  
[user@host ~]$ ssh-add .ssh/key-with-pass  
Enter passphrase for .ssh/key-with-pass: your_passphrase  
Identity added: .ssh/key-with-pass (user@host.lab.example.com)
```

다음 `ssh` 명령은 기본 개인 키 파일을 사용하여 원격 SSH 서버의 계정에 액세스합니다.

```
[user@host ~]$ ssh user@remotehost  
Last login: Mon Mar 14 06:51:36 2022 from host.example.com  
[user@remotehost ~]$
```

다음 `ssh` 명령은 `~/.ssh/key-with-pass` 개인 키를 사용하여 원격 서버의 계정에 액세스합니다. 이 예제의 개인 키는 이전에 암호가 해독되어 `ssh-agent` 키 관리자에 추가되었으므로 `ssh` 명령에서 개인 키의 암호를 해독하기 위한 암호를 요청하지 않습니다.

```
[user@host ~]$ ssh -i .ssh/key-with-pass user@remotehost  
Last login: Mon Mar 14 06:58:43 2022 from host.example.com  
[user@remotehost ~]$
```

`ssh-agent` 키 관리자를 사용한 세션에서 로그아웃하면 캐시된 모든 암호가 메모리에서 지워집니다.

기본 SSH 연결 문제 해결

키 쌍 인증을 사용한 원격 액세스에 성공하지 못하면 SSH가 복잡해 보일 수 있습니다. `ssh` 명령은 `-v`, `-vv`, `-vvv` 옵션을 통해 세 가지 세부 정보 표시 수준을 제공합니다. 각 옵션은 `ssh` 명령을 사용하는 동안 더 많은 양의 디버깅 정보를 제공합니다.

다음 예제에서는 가장 낮은 세부 정보 표시 옵션을 사용할 때 제공되는 정보를 보여줍니다.

```
[user@host ~]$ ssh -v user@remotehost  
OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021 ①  
debug1: Reading configuration data /etc/ssh/ssh_config ②  
debug1: Reading configuration data /etc/ssh/ssh_config.d/01-training.conf  
debug1: /etc/ssh/ssh_config.d/01-training.conf line 1: Applying options for *  
debug1: Reading configuration data /etc/ssh/ssh_config.d/50-redhat.conf  
...output omitted...  
debug1: Connecting to remotehost [192.168.1.10] port 22. ③  
debug1: Connection established.  
...output omitted...  
debug1: Authenticating to remotehost:22 as 'user' ④  
...output omitted...  
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password ⑤  
...output omitted...  
debug1: Next authentication method: publickey ⑥  
debug1: Offering public key: /home/user/.ssh/id_rsa RSA  
SHA256:hDVJjd7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ⑦
```

1장 | 시스템 액세스 및 지원 받기

```
debug1: Server accepts key: /home/user/.ssh/id_rsa RSA
SHA256:hDVJjD7xrUjXGZVRJQixxFV6NF/ssMjS6AuQ1+VqUc4 ⑧
Authenticated to remotehost ([192.168.1.10]:22) using "publickey".
...output omitted...
[user@remotehost ~]$
```

- ➊ OpenSSH 및 OpenSSL 버전입니다.
- ➋ OpenSSH 구성 파일입니다.
- ➌ 원격 호스트에 연결합니다.
- ➍ 원격 호스트에서 사용자 인증을 시도합니다.
- ➎ 원격 호스트에서 허용하는 인증 방법입니다.
- ➏ SSH 키를 사용하여 사용자 인증을 시도합니다.
- ➐ /home/user/.ssh/id_rsa 키 파일을 사용하여 인증합니다.
- ➑ 원격 호스트가 SSH 키를 수락합니다.

시도한 인증 방법이 실패하면 원격 SSH 서버는 사용 가능한 모든 방법이 시도될 때까지 다른 허용된 인증 방법으로 장애 복구합니다. 다음 예제에서는 SSH 키를 사용한 원격 액세스가 실패한 후 SSH 서버가 암호 인증을 제공하여 성공하는 과정을 보여줍니다.

```
[user@host ~]$ ssh -v user@remotehost
...output omitted...
debug1: Next authentication method: publickey
debug1: Offering public key: /home/user/.ssh/id_rsa RSA
SHA256:bsB6l5R184zvxNlrcRMmYd32oBkU1LgQj09dUBZ+Z/k
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password
...output omitted...
debug1: Next authentication method: password
user@remotehost's password: password
Authenticated to remotehost ([172.25.250.10]:22) using "password".
...output omitted...
[user@remotehost ~]$
```

SSH 클라이언트 구성

~/.ssh/config 파일을 생성하여 SSH 연결을 사전 구성할 수 있습니다. 구성 파일 내에서 특정 호스트의 사용자, 키, 포트와 같은 연결 매개 변수를 지정할 수 있습니다. 이 파일을 사용하면 호스트에 연결할 때마다 명령 매개 변수를 수동으로 지정할 필요가 없습니다. 서로 다른 사용자와 키로 두 개의 호스트 연결을 사전 구성하는 다음 ~/.ssh/config 파일을 살펴보십시오.

```
[user@host ~]$ cat ~/.ssh/config
host servera
    HostName          servera.example.com
    User              usera
    IdentityFile     ~/.ssh/id_rsa_servera

host serverb
```

HostName	serverb.example.com
User	userb
IdentityFile	~/.ssh/id_rsa_serverb



참조

`ssh-keygen(1)`, `ssh-copy-id(1)`, `ssh-agent(1)` 및 `ssh-add(1)` 도움말 페이지

▶ 연습 가이드

SSH 키 기반 인증 구성

이 연습에서는 SSH에 키 기반 인증을 사용하도록 사용자를 구성합니다.

결과

- 암호 보호를 사용하지 않고 SSH 키 쌍을 생성합니다.
- 암호 보호를 사용하여 SSH 키 쌍을 생성합니다.
- 암호 없는 SSH 키와 암호로 보호된 SSH 키를 둘 다 사용하여 인증합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start ssh-configure
```

지침

▶ 1. student 사용자로 serverb 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

▶ 2. serverb 시스템에서 operator1 사용자로 전환합니다. redhat을 암호로 사용합니다.

```
[student@serverb ~]$ su - operator1
Password: redhat
[operator1@serverb ~]$
```

▶ 3. SSH 키 세트를 생성합니다. 암호를 입력하지 마십시오.

```
[operator1@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator1/.ssh/id_rsa): Enter
Created directory '/home/operator1/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator1/.ssh/id_rsa.
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:JainiQdnRosC+xXh operator1@serverb.lab.example.com
```

```
The key's randomart image is:
+---[RSA 3072]----+
|E+*+ooo .          |
|.= o.o o .         |
|o.. = . . o        |
|+. + * . o .       |
|+= X . S +         |
| + @ + = .         |
| . + = o           |
| .o . . .          |
|o     o..          |
+---[SHA256]-----+
```

- ▶ 4. 암호로 **redhat**을 사용하여 **servera** 시스템의 **operator1** 사용자에게 SSH 키 쌍의 공개 키를 전송합니다.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/operator1/.ssh/id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:h/hEJa/anxp6AP7BmB5azIPVbPNqieh0oKi4KW0TK80.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
operator1@servera's password: redhat

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 5. 원격 대화형 쉘에 액세스하지 않고 **ssh** 명령을 사용하여 원격으로 **servera** 시스템에서 **hostname** 명령을 실행합니다.

```
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

앞의 **ssh** 명령은 내보낸 공개 키에 암호 없는 개인 키를 사용하여 **servera** 시스템에서 **operator1** 사용자로 인증하기 때문에 암호를 묻는 메시지를 표시하지 않습니다.

개인 키 파일에 대한 액세스 권한만 있으면 누구든지 **operator1** 사용자로 **servera** 시스템에 로그인할 수 있기 때문에 이 방법은 안전하지 않습니다.

이 연습의 다음 단계에서는 개인 키를 암호화하고 암호를 추가하여 개인 키에 대한 액세스를 보호하여 보안을 강화합니다.

- ▶ 6. 기본 이름을 사용하고 암호 없이 다른 SSH 키 세트를 생성하여 이전에 생성된 SSH 키 파일을 덮어씁니다. 새 SSH 키를 사용하여 **servera** 시스템에 연결해 봅니다. **ssh** 명령은 SSH 키로 인증할 수 없

으므로 암호를 요청합니다. **ssh** 명령에 **-v**(세부 정보 표시) 옵션을 사용해서 다시 실행하여 확인합니다.

servera 시스템의 **operator1** 사용자에게 SSH 키 쌍의 새 공개 키를 전송하여 이전 공개 키를 교체합니다. **servera** 시스템에서 **operator1** 사용자의 암호로 **redhat** 을 사용합니다. 원격 대화형 쉘에 액세스하지 않고 **ssh** 명령을 통해 원격으로 **servera** 시스템에서 **hostname** 명령을 실행하여 다시 작동하는지 확인합니다.

- 6.1. 다시 기본 이름을 사용하고 암호 없이 다른 SSH 키 세트를 생성하여 이전에 생성된 SSH 키 파일을 덮어씁니다.

```
[operator1@serverb ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/operator1/.ssh/id_rsa): Enter
/home/operator1/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/operator1/.ssh/id_rsa
Your public key has been saved in /home/operator1/.ssh/id_rsa.pub
...output omitted...
```

- 6.2. 새 SSH 키를 사용하여 **servera** 시스템에 연결해 봅니다. **ssh** 명령은 SSH 키로 인증할 수 없으므로 암호를 요청합니다. 암호를 입력하라는 메시지가 표시되면 **Ctrl+c**를 눌러 **ssh** 명령을 종료합니다. **ssh** 명령에 **-v**(세부 정보 표시) 옵션을 사용해서 다시 실행하여 확인합니다. 암호를 입력하라는 메시지가 표시되면 **Ctrl+c**를 다시 눌러 **ssh** 명령을 종료합니다.

```
[operator1@serverb ~]$ ssh operator1@servera hostname
operator1@servera's password: ^C
[operator1@serverb ~]$ ssh -v operator1@servera hostname
OpenSSH_8.7p1, OpenSSL 3.0.1 14 Dec 2021
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Reading configuration data /etc/ssh/ssh_config.d/01-training.conf
...output omitted...
debug1: Next authentication method: publickey
debug1: Offering public key: /home/operator1/.ssh/id_rsa RSA
SHA256:ad597zf64xckV26xht8bjQbzqSPu0XQPXksGEWVsP80
debug1: Authentications that can continue: publickey,gssapi-keyex,gssapi-with-mic,password
debug1: Trying private key: /home/operator1/.ssh/id_dsa
debug1: Trying private key: /home/operator1/.ssh/id_ecdsa
debug1: Trying private key: /home/operator1/.ssh/id_ecdsa_sk
debug1: Trying private key: /home/operator1/.ssh/id_ed25519
debug1: Trying private key: /home/operator1/.ssh/id_ed25519_sk
debug1: Trying private key: /home/operator1/.ssh/id_xmss
debug1: Next authentication method: password
operator1@servera's password: ^C
```

- 6.3. **servera** 시스템의 **operator1** 사용자에게 SSH 키 쌍의 새 공개 키를 전송하여 이전 공개 키를 교체합니다. **servera** 시스템에서 **operator1** 사용자의 암호로 **redhat** 을 사용합니다. 원격 대화형 쉘에 액세스하지 않고 **ssh** 명령을 통해 원격으로 **servera** 시스템에서 **hostname** 명령을 실행하여 다시 작동하는지 확인합니다.

```
[operator1@serverb ~]$ ssh-copy-id operator1@servera
...output omitted...
operator1@servera's password: redhat

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
[operator1@serverb ~]$ ssh operator1@servera hostname
servera.lab.example.com
```

- ▶ 7. 암호로 보호된 다른 SSH 키 세트를 생성합니다. 키를 /home/operator1/.ssh/key2로 저장합니다. **redhatpass**를 개인 키의 암호로 사용합니다.

```
[operator1@serverb ~]$ ssh-keygen -f .ssh/key2
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase): redhatpass
Enter same passphrase again: redhatpass
Your identification has been saved in .ssh/key2.
Your public key has been saved in .ssh/key2.pub.
The key fingerprint is:
SHA256:0CtCjfPm5QrbPBgqb operator1@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]---+
|O=X*          |
|OB=.          |
|E*o.          |
|Booo .        |
|..= . o S    |
|+.o  o        |
|+.oo+ o       |
|+o.O.+        |
|+. . =o.      |
+---[SHA256]---+
```

- ▶ 8. 암호로 보호된 키 쌍의 공개 키를 servera 시스템의 operator1 사용자에게 전송합니다. 이 명령은 이전 단계에서 servera 시스템에 내보낸 암호 없는 개인 키의 공개 키를 사용하기 때문에 암호를 묻는 메시지를 표시하지 않습니다.

```
[operator1@serverb ~]$ ssh-copy-id -i .ssh/key2.pub operator1@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/key2.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'operator1@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- ▶ 9. ssh 명령을 사용하여 원격으로 servera 시스템에서 hostname 명령을 실행합니다. /home/operator1/.ssh/key2 키를 ID 파일로 사용합니다. 앞의 단계에서 개인 키에 대해 설정한 redhatpass를 암호로 지정합니다.

이 명령은 SSH 키 쌍의 개인 키를 보호하는 데 사용한 암호를 묻는 메시지를 표시합니다. 공격자가 개인 키에 대한 액세스 권한을 얻더라도 개인 키 자체가 암호로 보호되기 때문에 개인 키를 사용하여 다른 시스템에 액세스할 수 없습니다. ssh 명령은 servera 시스템의 operator1 사용자와 다른 암호를 사용하므로 사용자가 두 개 암호를 모두 알아야 합니다.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
Enter passphrase for key '.ssh/key2': redhatpass
servera.lab.example.com
```

SSH를 사용하여 로그인하는 동안 암호를 대화형으로 입력하지 않으려면 다음 단계와 같이 ssh-agent 프로그램을 사용합니다. 관리자가 원격 시스템에 정기적으로 로그인하는 경우 ssh-agent 프로그램을 사용하면 더 편리하고 안전합니다.

- ▶ 10. Bash 쉘에서 ssh-agent 프로그램을 실행하고 쉘 세션에 SSH 키 쌍의 암호로 보호된 개인 키(/home/operator1/.ssh/key2)를 추가합니다.

이 명령은 ssh-agent 프로그램을 시작하고 해당 프로그램을 사용하도록 쉘 세션을 구성합니다. 그런 다음 ssh-add 명령을 사용하여 잠금 해제된 개인 키를 ssh-agent 프로그램에 제공합니다.

```
[operator1@serverb ~]$ eval $(ssh-agent)
Agent pid 1729
[operator1@serverb ~]$ ssh-add .ssh/key2
Enter passphrase for .ssh/key2: redhatpass
Identity added: .ssh/key2 (operator1@serverb.lab.example.com)
```

- ▶ 11. 원격 대화형 쉘에 액세스하지 않고 원격으로 servera 시스템에서 hostname 명령을 실행합니다. /home/operator1/.ssh/key2 키를 ID 파일로 사용합니다.

이 명령은 대화형으로 암호를 입력하라는 메시지를 표시하지 않습니다.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera hostname
servera.lab.example.com
```

- ▶ 12. workstation 시스템에서 다른 터미널을 열고 student 사용자로 serverb 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- ▶ 13. serverb 시스템에서 operator1 사용자로 전환하고 servera 시스템에 원격으로 로그인합니다. /home/operator1/.ssh/key2 키를 ID 파일로 사용하여 SSH 키로 인증합니다.

13.1. su 명령을 사용하여 operator1 사용자로 전환합니다. operator1 사용자의 암호로 redhat을 사용합니다.

```
[student@serverb ~]$ su - operator1
Password: redhat
[operator1@serverb ~]$
```

13.2. operator1 사용자로 servera 시스템에 로그인합니다.

이 명령은 **ssh-agent** 프로그램을 시작한 쉘과 동일한 쉘에서 SSH 연결을 호출하지 않았기 때문에 대화형으로 암호를 입력하라는 메시지를 표시합니다.

```
[operator1@serverb ~]$ ssh -i .ssh/key2 operator1@servera
Enter passphrase for key '.ssh/key2': redhatpass
...output omitted...
[operator1@servera ~]$
```

▶ 14. 추가 터미널을 모두 종료하고 닫은 다음, workstation 시스템으로 돌아갑니다.

14.1. 추가 터미널 창을 종료하고 닫습니다. **exit** 명령은 **operator1** 사용자 쉘을 종료하고, **ssh-agent**가 활성화된 쉘 세션을 종료한 다음, **serverb** 시스템의 **student** 사용자 쉘로 돌아갑니다.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[operator1@serverb ~]$
```

14.2. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[operator1@serverb ~]$ exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish ssh-configure
```

이것으로 섹션을 완료합니다.

진단 보고서 생성

목표

Red Hat Customer Portal 주요 리소스를 설명하고 사용하여 Red Hat 설명서 및 Knowledgebase에서 정보를 찾습니다.

Red Hat Customer Portal의 리소스

Red Hat Customer Portal(<https://access.redhat.com>)에서 고객은 설명서, 다운로드, 툴, 기술 전문 지식에 액세스할 수 있습니다. 고객은 Knowledgebase를 통해 솔루션, FAQ, 문서를 검색할 수 있습니다. 다음 목록은 Red Hat Customer Portal의 몇 가지 기능을 보여줍니다.

- 공식 제품 설명서, 솔루션, FAQ에 액세스합니다.
- 지원 사례를 제출하고 관리합니다.
- 소프트웨어 구독 및 권한 부여를 관리합니다.
- 소프트웨어 다운로드, 업데이트 및 평가를 받습니다.
- Red Hat 제품에 대한 보안 권고 카탈로그에 액세스합니다.
- Red Hat 리소스에 대한 통합 검색 엔진에 액세스합니다.
- 백서, 정보 시트, 멀티미디어 프레젠테이션에 액세스합니다.
- 커뮤니티 토론에 참여합니다.

사이트 일부는 모든 사용자가 공개적으로 액세스할 수 있지만 활성 서브스크립션의 필요한 영역도 있습니다. Red Hat Customer Portal에 액세스하는 데 도움이 필요하면 <https://access.redhat.com/help/> 를 방문하십시오.

Red Hat Customer Portal 둘러보기

<https://access.redhat.com/> 을 방문하여 Red Hat Customer Portal에 액세스합니다. 이 섹션에서는 <https://access.redhat.com/start>에 있는 Red Hat Customer Portal 둘러보기를 소개합니다.

둘러보기를 통해 포털 기능을 살펴보고 Red Hat 서브스크립션의 이점을 극대화할 수 있습니다. Red Hat Customer Portal에 로그인한 후 **Tour the Customer Portal** 버튼을 클릭합니다.

WELCOME TO THE RED HAT CUSTOMER PORTAL 창이 표시됩니다. **Let's go** 버튼을 클릭하여 둘러보기를 시작합니다.

상단 네비게이션 바

둘러보기의 첫 번째 메뉴는 상단 네비게이션 바에 있으며 Subscriptions(서브스크립션), Downloads(다운로드), Containers(컨테이너), Support Cases(지원 사례)입니다.

Subscriptions 메뉴는 등록된 시스템, 서브스크립션 및 자격을 관리할 수 있는 새 페이지를 엽니다. 이 페이지에는 적용 가능한 애라타 정보가 표시됩니다. 시스템을 등록하고 올바른 자격을 보장하기 위해 활성화 키를 생성할 수 있습니다. 계정의 조직 관리자가 이 페이지에 대한 액세스를 제한할 수도 있습니다.

Downloads 메뉴는 제품의 다운로드에 액세스하고 자격 없는 제품의 평가판을 요청할 수 있는 새 페이지를 엽니다.

Support Cases 메뉴는 조직에서 권한을 부여한 경우 사례 관리 시스템을 통해 지원 사례를 생성, 추적, 관리할 수 있는 새 페이지를 엽니다.

1장 | 시스템 액세스 및 지원 받기

User Menu 메뉴를 사용하여 본인 계정과 조직 관리자로 담당하는 계정, 본인 프로필, 이메일 알림 옵션을 관리합니다.

지구본 아이콘은 Red Hat Customer Portal의 언어 기본 설정을 지정할 수 있는 **Language** 메뉴를 엽니다.

Red Hat Customer Portal 메뉴 탐색

기본 페이지의 상단 네비게이션 바 아래에는 사이트의 주요 리소스 범주로 이동할 수 있는 메뉴가 있습니다.

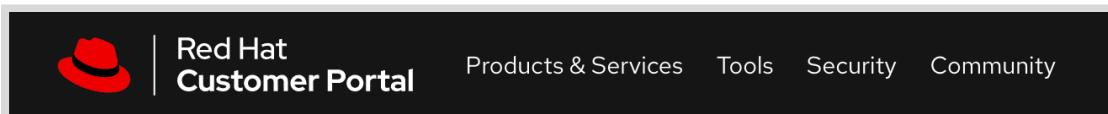


그림 1.2: Red Hat Customer Portal 메뉴

Products & Services 메뉴에서는 Product Hubs(제품 허브)에 액세스하여 제품별 평가, 시작하기 가이드, 기타 제품 지원 정보를 확인할 수 있습니다. Red Hat 제품 설명서, 지원 문서 기술 자료, 지원 정책에 액세스하고 Red Hat 지원에 문의할 수도 있습니다. 컨설팅, 기술 계정 관리, 교육 및 인증과 같은 Red Hat에서 제공하는 서비스에 액세스할 수 있습니다.

Tools 메뉴에서는 Red Hat 제품으로 성공할 수 있도록 지원하는 툴을 제공합니다. 툴은 제품 문제를 해결하는데 도움이 되며, 패키지 및 에라타 정보를 제공합니다. **Customer Portal Labs** 섹션에는 성능을 개선하고, 문제를 진단하고, 보안 문제를 식별하고, 구성을 최적화하는데 도움이 되는 웹 기반 애플리케이션 및 툴 컬렉션이 있습니다. **Red Hat Insights** 섹션은 플랫폼 및 애플리케이션을 분석하여 위험을 예측하고, 권장 조치를 수행하고, 비용을 추적하여 하이브리드 클라우드 환경을 관리하는데 도움이 됩니다. Insights는 중단이 발생하기 전이나 보안 이벤트 또는 과다 지출에 대해 관리자에게 경고합니다.

Security 메뉴에서는 Red Hat 제품 보안 센터에 액세스하여 보안 업데이트를 가져오고 환경이 보안 취약점에 노출되지 않도록 방지할 수 있습니다. 이 섹션은 통해 중요한 보안 문제에 대한 정보를 확인하고 보안 권고, Red Hat CVE(Common Vulnerabilities and Exposures) 데이터베이스, 보안 랩, Red Hat 보안 블로그, 보안 측정, 심각도 등급, 백포팅 정책, 제품 서명 GPG(GNU Privacy Guard) 키에 액세스할 수 있습니다.

Community 메뉴에서는 **Customer Portal Community** 섹션에 액세스하여 토론 및 비공개 그룹을 사용할 수 있습니다. 이 섹션은 Red Hat 전문가, 고객 및 파트너가 소통하고 협업할 수 있는 곳입니다. 이 섹션에는 토론 포럼, 블로그, 예정된 이벤트에 대한 정보가 포함되어 있습니다.



참고

How to Personalize Your Customer Portal experience 메뉴, Explore the Benefits of Your Red Hat subscription 메뉴, How to Engage Red Hat Support 메뉴의 섹션을 포함하여 Red Hat 사용 시작 [<https://access.redhat.com/start>]에서 전체 둘러보기를 수행하는 것이 좋습니다. 이러한 서브스크립션 리소스에 액세스하려면 활성 서브스크립션이 필요합니다.

Red Hat 고객 지원에 문의

Red Hat Customer Portal에서는 활성 서브스크립션 있는 고객에게 기술 지원을 제공합니다. 지원 사례 또는 채팅 세션을 열거나 전화를 걸어 지원에 문의할 수 있습니다. 자세한 내용을 보려면 https://access.redhat.com/support/policy/support_process 주소를 방문하십시오.

지원 사례 준비

Red Hat 지원에 문의하기 전에 보고서와 관련된 정보를 수집하는 것이 중요합니다.

문제를 명확하게 정의합니다. 문제와 증상을 구체적으로 설명합니다. 문제를 재현하기 위한 자세한 단계를 제공합니다.

1장 | 시스템 액세스 및 지원 받기

배경 정보를 수집합니다. 어떤 제품과 버전이 영향을 받습니까? 관련 진단 정보를 제공할 준비가 됩니다. 이 정보에는 **sos report** 명령의 출력이 포함될 수 있습니다. 커널 문제의 경우 정보가 시스템의 **kdump** 크래시 덤프나 충돌한 시스템 모니터에 표시된 커널 역추적의 디지털 사진으로 구성될 수도 있습니다.

심각도 수준을 결정합니다. Red Hat은 네 가지 심각도 수준을 사용하여 문제를 분류합니다. 긴급 및 높음 심각도의 문제 보고서의 경우 후속 조치로 해당 지역 지원 센터에 전화로 문의해야 합니다(<https://access.redhat.com/support/contact/technicalSupport> 참조).

심각도	설명
긴급(심각도 1)	운영 환경에서 소프트웨어 사용에 심각한 영향을 미치는 문제입니다. 이 심각도에는 프로덕션 데이터 손실 또는 프로덕션 시스템 오작동이 포함됩니다. 이 경우 비즈니스 운영이 중단되며, 대안으로 수행할 수 있는 조치가 없습니다.
높음(심각도 2)	소프트웨어가 작동하지만 프로덕션 환경에서 소프트웨어 사용이 심각하게 감소하는 문제입니다. 이 경우 비즈니스 운영이 큰 영향을 받으며, 대안으로 수행할 수 있는 조치가 없습니다.
중간(심각도 3)	프로덕션 환경이나 개발 환경에서 소프트웨어 사용이 심각하지는 않지만 부분적으로 손실되는 문제입니다. 프로덕션 환경에서 비즈니스는 이 문제로 인해 중간~낮은 수준의 영향을 받습니다. 대안으로 수행할 수 있는 조치를 사용하면 비즈니스가 계속 운영됩니다. 개발 환경에서는 이 경우 프로젝트를 프로덕션 환경으로 마이그레이션하는 데 문제가 발생합니다.
낮음(심각도 4)	일반적인 사용에 관한 질문으로, 문서상의 오류를 보고하거나 향후 제품 개선이나 수정에 관한 권장 사항을 언급합니다. 비즈니스 또는 시스템 성능이나 기능은 이 문제로 인해 낮은 수준의 영향을 받거나 영향을 받지 않습니다. 개발 환경에서 비즈니스는 이 문제로 인해 중간~낮은 수준의 영향을 받지만, 대안으로 수행할 수 있는 조치를 사용하면 비즈니스가 계속 운영됩니다.

sos 보고서 유ти리티

일반적으로 **sos** 보고서는 Red Hat 기술 지원에서 보고된 문제를 조사하기 위한 시작점입니다. 이 유ти리티는 Red Hat 기술 지원에서 보고된 문제를 조사하는 데 필요한 진단 정보를 수집하는 표준화된 방법을 제공합니다. **sos report** 명령은 하나 이상의 시스템에서 다양한 디버깅 정보를 수집하며, 중요한 데이터를 제거하는 옵션을 제공합니다. 이 보고서는 Red Hat 지원 사례에 첨부됩니다. **sos collect** 명령은 지정된 노드 세트에서 개별 **sos** 보고서를 실행하고 수집합니다. **sos clean** 명령은 사용자 이름, 호스트 이름, IP 또는 MAC 주소, 기타 사용자 지정 데이터와 같은 잠재적으로 중요한 정보를 난독화합니다.

다음 목록에는 보고서에 수집될 수 있는 정보가 나와 있습니다.

- 실행 중인 커널 버전
- 로드된 커널 모듈
- 시스템 및 서비스 구성 파일
- 진단 명령 출력
- 설치된 패키지 목록

웹 콘솔 또는 명령줄을 사용하여 Red Hat 기술 지원에 제출할 진단 보고서를 생성할 수 있습니다.

웹 콘솔을 사용하여 sos 보고서 생성

웹 콘솔을 사용하여 **sos** 보고서를 생성하려면 권한 있는 사용자로 로그인해야 합니다.

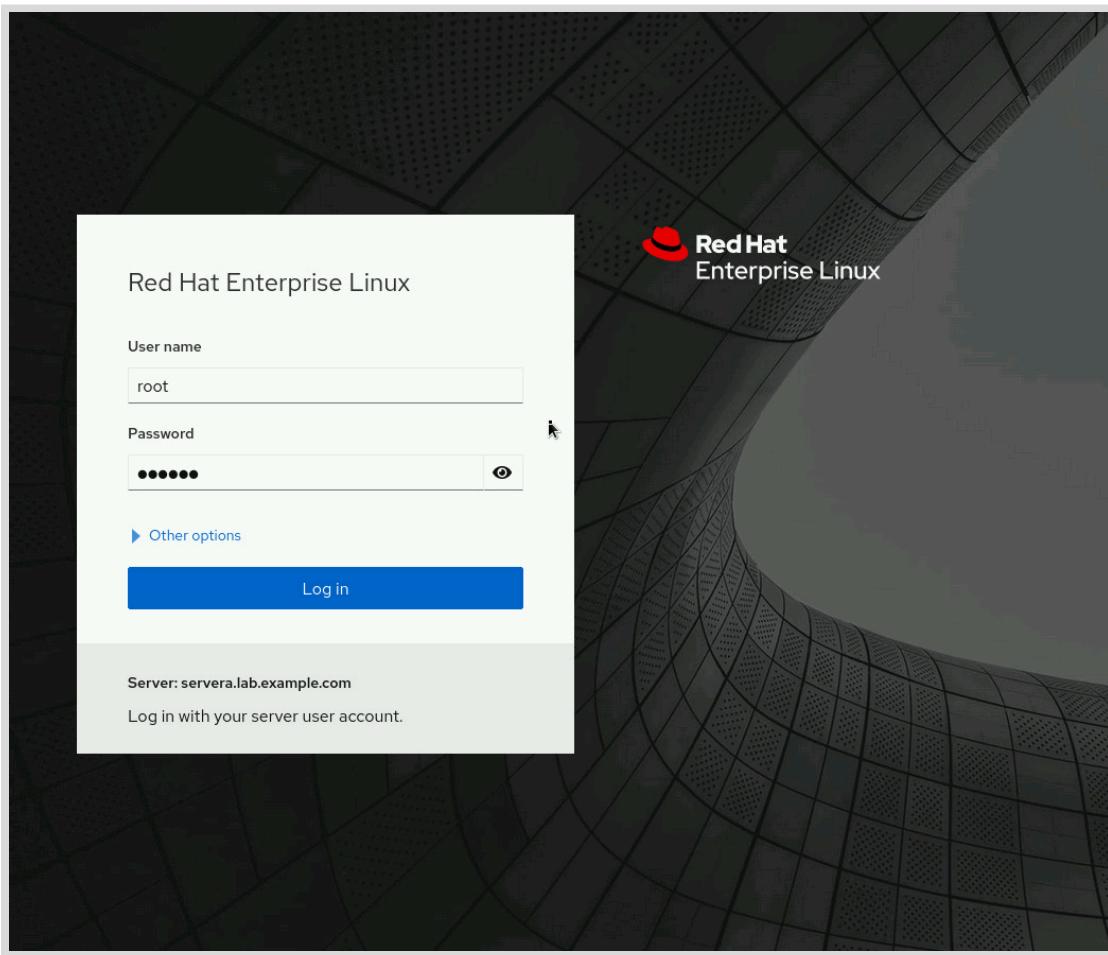


그림 1.3: 권한 있는 사용자를 사용하여 로그인

Diagnostic Reports 를 클릭한 다음 Create report를 클릭합니다.

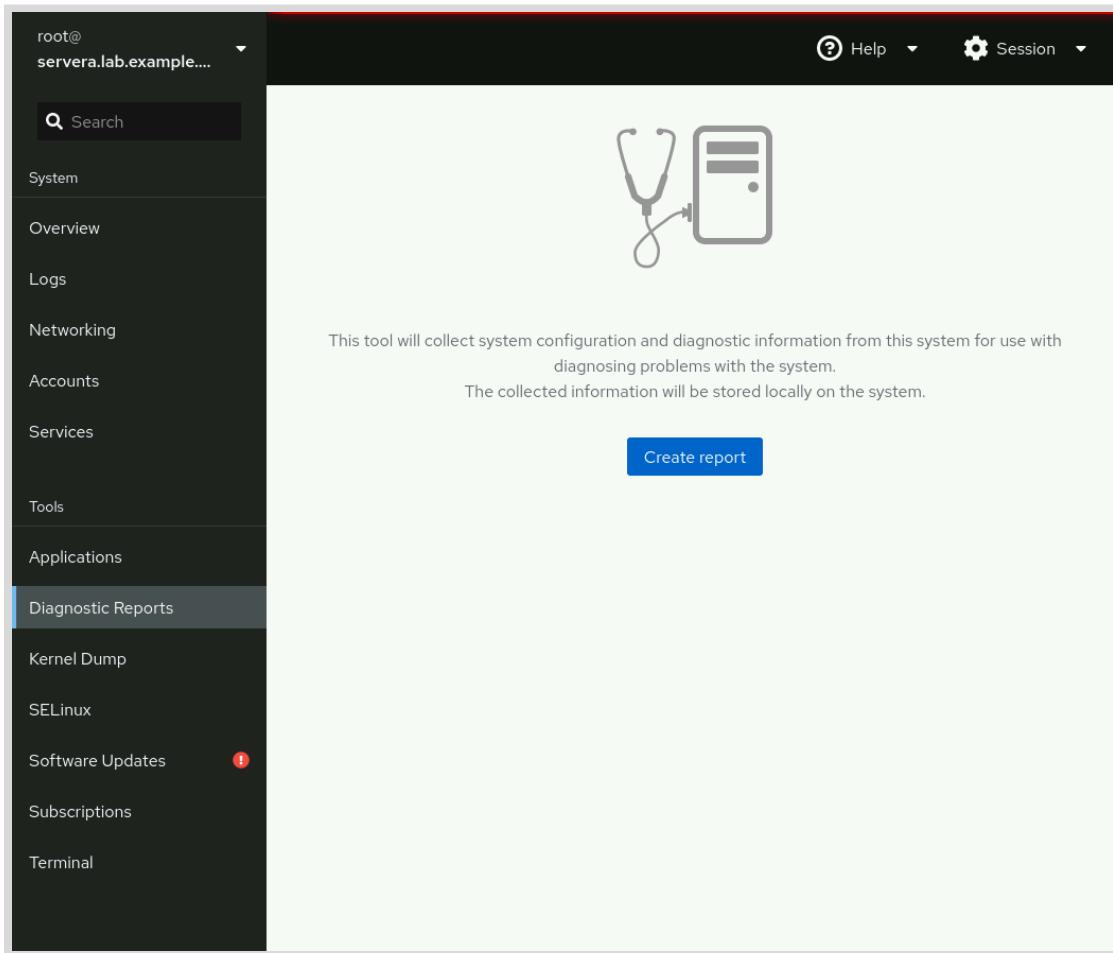


그림 1.4: 진단 보고서 생성

진단 보고서를 생성하는 데 몇 분이 걸립니다.

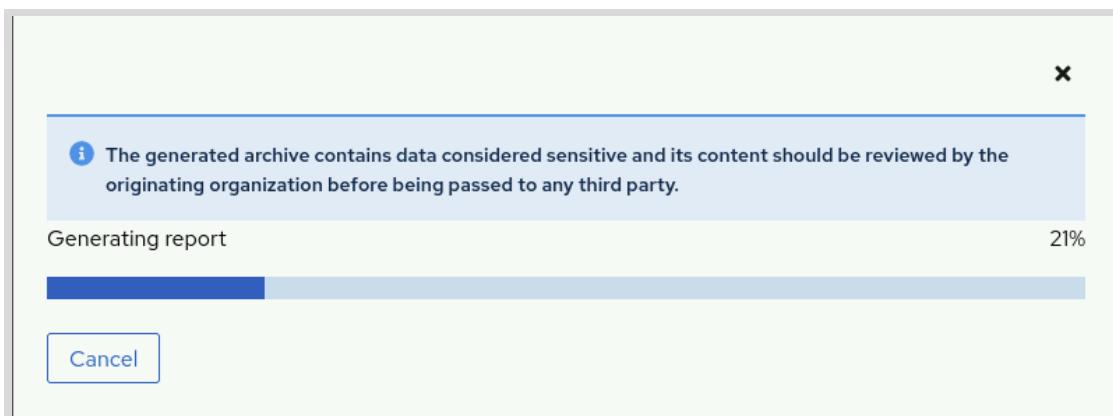


그림 1.5: 진단 보고서 생성

Download report 를 클릭한 다음 Save File 을 클릭하여 진단 보고서를 저장합니다.



그림 1.6: 진단 보고서를 로컬 시스템에 다운로드

명령줄을 사용하여 sos 보고서 생성

Red Hat Enterprise Linux는 **sos** 패키지를 사용하여 **sos** 보고서 유ти리티를 설치합니다.

```
[root@host ~]# dnf install sos
...output omitted...
Complete!
```

sos 보고서를 생성하려면 루트 권한이 필요합니다. **sos report** 명령을 실행하여 보고서를 생성합니다.

```
[root@host ~]# sos report
...output omitted...
Press ENTER to continue, or CTRL-C to quit.

 Optionally, please enter the case id that you are generating this report for []:
...output omitted...
Your sosreport has been generated and saved in:
 /var/tmp/sosreport-host-2022-03-29-wixbhpz.tar.xz
...output omitted...
Please send this file to your support representative.
```

앞의 명령에서 지원 사례 ID를 제공하면 이전에 생성한 지원 사례에 보고서가 직접 첨부됩니다. **sos report** 명령의 **--utility** 옵션을 사용하여 기술 지원에 보고서를 전송할 수도 있습니다.

sos report 명령이 이전 위치에 아카이브 파일을 생성했는지 확인합니다.

```
[root@host ~]# ls -l /var/tmp/
total 9388
-rw----- 1 root root 9605952 Mar 29 02:09 sosreport-host-2022-03-29-
wixbhpz.tar.xz
-rw-r--r-- 1 root root       65 Mar 29 02:09 sosreport-host-2022-03-29-
wixbhpz.tar.xz.sha256
...output omitted...
```

sos clean 명령은 보고서에서 개인 정보를 난독화합니다.

```
[root@host ~]# sos clean /var/tmp/sosreport-host-2022-03-29-wixbhpz.tar.xz*
...output omitted...
Press ENTER to continue, or CTRL-C to quit.
...output omitted...
The obfuscated archive is available at
    /var/tmp/sosreport-host0-2022-03-29-wixbhpz-obfuscated.tar.xz
...output omitted...
Please send the obfuscated archive to your support representative and keep the
mapping file private
```

Red Hat 기술 지원에 **sos** 보고서 전송

Red Hat 기술 지원에 **sos** 보고서를 전송하려면 다음 방법 중 하나를 선택합니다.

- **sos report** 명령의 **--upload** 옵션을 사용하여 **sos** 보고서를 전송합니다.
- 지원 사례에 첨부하여 Red Hat Customer Portal에 **sos** 보고서를 전송합니다.

Red Hat 개발자 프로그램에 참여

Red Hat 개발자 프로그램(<https://developers.redhat.com>)은 개발 목적의 Red Hat 소프트웨어, 설명서 및 마이크로 서비스, 서비스 컴퓨팅, Kubernetes, Linux 관련 전문가의 프리미엄 도서에 대한 서브스크립션 자격을 제공합니다. 예정된 이벤트 및 교육 관련 정보와 기타 유용한 리소스에 대한 블로그 링크도 사용할 수 있습니다.

자세한 내용은 <https://developers.redhat.com/>을 참조하십시오.



참조

[sosreport\(1\)](#) 도움말 페이지

Red Hat 기술 지원에 문의

https://access.redhat.com/support/policy/support_process/

도움말 - Red Hat Customer Portal

<https://access.redhat.com/help/>

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/getting_the_most_from_your_support_experience/generating-an-sos-report-for-technical-support_getting-the-most-from-your-support-experience
에서 Generating an SOS Report for Technical Support를 참조하십시오.

▶ 연습 가이드

진단 보고서 생성

이 연습에서는 웹 콘솔을 사용하여 지원 사례의 일부로 Red Hat Customer Portal에 제출할 수 있는 진단 보고서를 생성합니다.

결과

- 지원 사례의 일부로 Red Hat Customer Portal에 제출할 수 있는 진단 보고서를 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start support-portal
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
Warning: Permanently added 'servera' (ED25519) to the list of known hosts.
Activate the web console with: systemctl enable --now cockpit.socket
...output omitted...
[student@servera ~]$
```

- ▶ 2. **cockpit** 서비스를 시작합니다.

```
[student@servera ~]$ systemctl start cockpit.socket
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to start 'cockpit.socket'.
Authenticating as: Student User (student)
Password: student
==== AUTHENTICATION COMPLETE ====
[student@servera ~]$
```

- ▶ 3. **cockpit** 서비스의 상태를 확인합니다.

```
[student@servera ~]$ systemctl status cockpit.socket
● cockpit.socket - Cockpit Web Service Socket
   Loaded: loaded (/usr/lib/systemd/system/cockpit.socket; disabled; vendor
   preset: disabled)
     Active: active (listening) since Mon 2022-03-28 01:41:13 EDT; 1min 27s ago
       Until: Mon 2022-03-28 01:41:13 EDT; 1min 27s ago
     Triggers: ● cockpit.service
    Docs: man:cockpit-ws(8)
```

```
Listen: [::]:9090 (Stream)
...output omitted...
Mar 28 01:41:13 servera.lab.example.com systemd[1]: Starting Cockpit Web Service
Socket...
Mar 28 01:41:13 servera.lab.example.com systemd[1]: Listening on Cockpit Web
Service Socket.
```

▶ 4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

▶ 5. **workstation** 시스템에서 Firefox 웹 브라우저를 열고 **servera.lab.example.com** 주소에서 실행 중인 웹 콘솔 인터페이스에 로그인합니다. **root**을 암호로 사용하여 **redhat** 사용자로 로그인합니다.

- 5.1. Firefox 웹 브라우저를 열고 **https://servera.lab.example.com:9090** 주소로 이동합니다.
- 5.2. 메시지가 표시되면 자체 서명된 인증서를 예외로 추가하여 수락합니다.
- 5.3. **root**을 암호로 사용하여 **redhat** 사용자로 로그인합니다. 이제 진단 보고서를 작성하는데 필요한 권한 있는 사용자로 로그인되었습니다.
- 5.4. 왼쪽 네비게이션 창에서 **Diagnostic Reports** 메뉴를 클릭합니다. **Create Report** 버튼을 클릭합니다. 보고서를 작성하는 데 몇 분이 걸립니다.

▶ 6. 보고서가 준비되면 **Download report** 버튼을 클릭하여 파일을 저장합니다.

- 6.1. **Download report** 버튼을 클릭하고 **Save File** 버튼을 클릭합니다.
- 6.2. 웹 콘솔 세션에서 로그아웃하고 Firefox 웹 브라우저를 닫습니다.

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish support-portal
```

이것으로 섹션을 완료합니다.

Red Hat Insights를 사용하여 문제 탐지 및 해결

목표

Red Hat Insights를 사용하여 서버의 문제를 분석하고 문제를 수정하거나 해결한 다음, 솔루션이 성공했는지 확인합니다.

Red Hat Insights 소개

Red Hat Insights는 인프라에서 Red Hat 제품을 실행하는 시스템의 보안, 성능, 가용성 및 안정성에 대한 위협을 식별하고 해결하는데 도움이 되는 예측 분석 툴입니다. Red Hat Insights는 SaaS(서비스로서의 소프트웨어) 제품으로 제공되므로 추가적인 인프라 요구 사항 없이 배포하고 확장할 수 있습니다. 또한 배포된 시스템에 적용되는, Red Hat의 최신 권장 사항과 업데이트를 활용할 수 있습니다.

Red Hat은 일반적인 지원 위험, 보안 취약점, 알려진 잘못된 구성 및 Red Hat에서 식별한 기타 문제를 기준으로 기술 자료를 주기적으로 업데이트합니다. Red Hat은 이러한 문제를 완화하거나 해결하는 작업을 검증하고 확인합니다. 이 지원을 활용하면 문제가 더 커지기 전에 사전에 문제를 식별하고, 우선 순위를 정하고, 해결할 수 있습니다.

탐지된 각 문제에 대해 Red Hat Insights는 위험 추정치와 문제를 완화하거나 해결하는 방법에 대한 권장 사항을 제공합니다. 권장 사항은 Ansible 플레이북과 같은 자료를 제안하거나 문제 해결에 도움이 되는 단계별 지침을 제공할 수 있습니다.

Red Hat Insights는 서비스에 등록된 각 시스템에 맞게 권장 사항을 조정합니다. Red Hat Insights 사용을 시작하려면 각 클라이언트 시스템에 에이전트를 설치하여 시스템의 런타임 구성에 대한 메타데이터를 수집합니다. 이 데이터는 지원 티켓을 해결하기 위해 `sosreport` 명령을 사용하여 Red Hat 지원에 제공할 수 있는 정보의 하위 집합입니다.

클라이언트 시스템이 전송하는 데이터를 제한하거나 난독화할 수 있습니다. 데이터를 제한하면, 제한하는 정보에 따라 일부 분석 규칙이 작동하지 않도록 차단할 수 있습니다.

서버를 등록하고 초기 시스템 메타데이터 동기화가 완료되면 Red Hat Cloud Portal의 Insights 콘솔에서 해당 서버와 모든 권장 사항을 볼 수 있습니다.

Red Hat Insights는 현재 다음 Red Hat 제품에 대한 예측 분석 및 권장 사항을 제공합니다.

- Red Hat Enterprise Linux 6.4 이상
- Red Hat Virtualization
- Red Hat Satellite 6 이상
- Red Hat OpenShift Container Platform
- Red Hat OpenStack Platform 7 이상
- Red Hat Ansible Automation Platform

Red Hat Insights 아키텍처 설명

Red Hat Insights에 시스템을 등록하면 시스템은 현재 구성에 관한 메타데이터를 Red Hat Insights 플랫폼에 즉시 전송합니다. 등록 후에는 Red Hat Insights에 제공한 메타데이터를 주기적으로 업데이트합니다. 시스템은 전송 중에 보호하기 위해 TLS 암호화를 사용하여 메타데이터를 전송합니다.

Red Hat 인사이트 플랫폼에서 수신한 데이터를 분석하고 결과를 <https://console.redhat.com/insights> 사이트에 표시합니다.

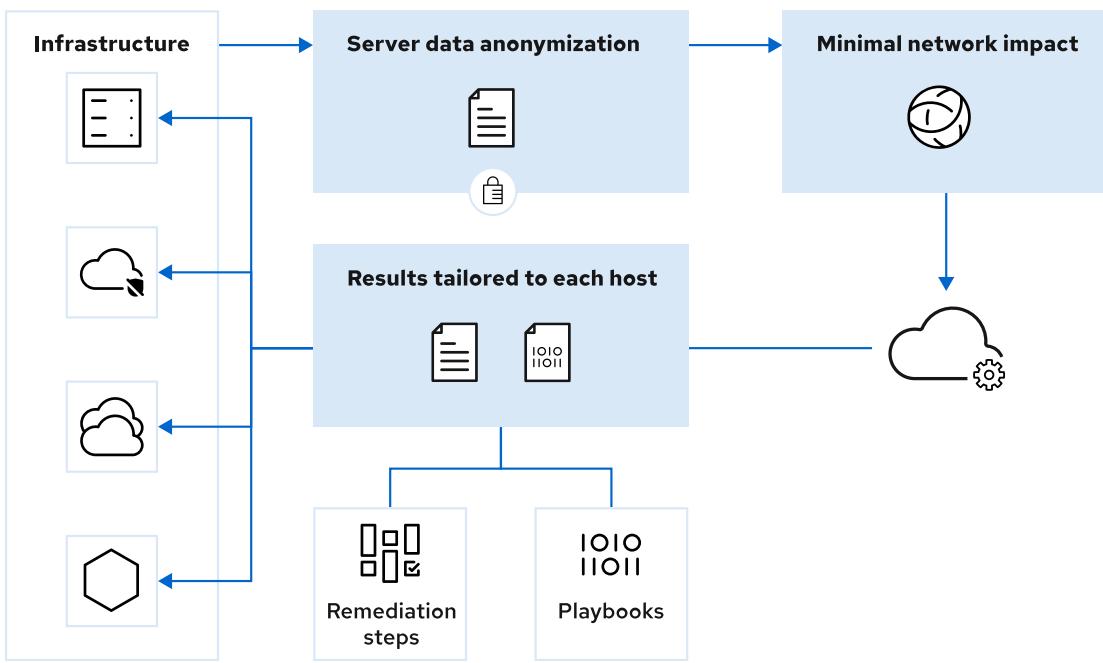


그림 1.7: Insights 고급 아키텍처

Red Hat Insights 클라이언트 설치

Red Hat Enterprise Linux 9에는 Insights가 서브스크립션의 일부로 포함되어 있습니다. 이전 버전의 Red Hat Enterprise Linux 서버를 사용하는 경우 시스템에 `insights-client` 패키지를 설치해야 합니다.

Red Hat Enterprise Linux 7.5부터 `redhat-access-insights` 패키지가 `insights-client` 패키지로 대체되었습니다. 다음 섹션에서는 `insights-client` 패키지를 설치하고 Red Hat 인사이트에 시스템을 등록하는 방법을 자세히 설명합니다.

Insights 클라이언트는 Insights에 제공된 메타데이터를 주기적으로 업데이트합니다. `insights-client` 명령을 사용하여 클라이언트 메타데이터를 새로 고칩니다.

```
[root@host ~]# insights-client
Starting to collect Insights data for host.example.com
Uploading Insights data.
Successfully uploaded report from host.example.com to account 1460291.
View details about this system on console.redhat.com:
https://console.redhat.com/insights/inventory/dc480efd-4782-417e-a496-cb33e23642f0
```

Red Hat Insights에 RHEL 시스템 등록

Red Hat Insights에 RHEL 서버를 신속하게 등록할 수 있습니다.

Red Hat 서브스크립션 관리 서비스를 사용하여 대화형으로 시스템을 등록합니다.

```
[root@host ~]# subscription-manager register --auto-attach
```

시스템에 `insights-client` 패키지가 설치되어 있는지 확인합니다. RHEL 8 이상 시스템에는 패키지가 기본적으로 설치되어 있습니다.

```
[root@host ~]# dnf install insights-client
```

insights-client --register 명령을 사용하여 시스템을 Insights 서비스에 등록하고 초기 시스템 메타데이터를 업로드합니다.

```
[root@host ~]# insights-client --register
```

Red Hat 인사이트(<https://console.redhat.com/insights>)에서 로그인되어 있고 웹 UI의 **Inventory** 섹션에 시스템이 표시되는지 확인합니다.

Name	Tags	OS	Last seen
serverb.lab.example.com		RHEL 9.1	Just now
servera.lab.example.com		RHEL 9.1	3 minutes ago
workstation.lab.example.com		RHEL 9.1	10 minutes ago

그림 1.8: 클라우드 포털의 Insights 인벤토리

Red Hat Insights 콘솔 탐색

Red Hat 인사이트는 <https://console.redhat.com/insights> 웹사이트에서 웹 브라우저를 통해 액세스할 수 있는 서비스 제품군을 제공합니다.

어드바이저 서비스를 사용하여 구성 문제 탐지

어드바이저 서비스에서는 시스템에 영향을 주는 구성 문제를 보고합니다. **Advisor > Recommendations** 메뉴에서 서비스에 액세스할 수 있습니다.

Name	Added	Category	Total risk	Risk of change	Systems	Ansible
"SMBLoria" Samba denial of service with externally listening process	5 years ago	Security	Important	Very Low	1	Yes
Decreased security: Adobe Flash Player installed	6 months ago	Security	Moderate	Moderate	1	Yes
Decreased security: Yum GPG verification disabled (third-party repos)	4 years ago	Security	Important	Very Low	1	No
Traffic occurs or services are allowed unexpectedly when firewall zone drifting is enabled	2 years ago	Availability	Moderate	Moderate	1	No
Performance degradation of I/O when commands timeout due to faulty storage hardware	5 years ago	Stability	Important	Very Low	0	No
D-Bus timeout occurs when there are a large number of existing abrt crash directories	2 years ago	Availability	Moderate	Moderate	0	Yes
Asynchronous I/O related operations fail when kernel parameter fs aio-max-nr limit is reached	3 years ago	Performance	Moderate	Low	0	No
Low density nodes detected	3 years ago	Performance	Moderate	Low	0	No
Running workload on nodes of an OpenShift cluster with an over-provisioned instance type size can result in cost increase	3 years ago	Performance	Moderate	Low	0	No

그림 1.9: 어드바이저 서비스의 권장 사항

문제가 발생할 때마다 Red Hat Insights는 문제를 파악하고, 문제 해결 작업에 우선 순위를 정하고, 사용 가능한 완화 또는 해결 방법을 확인하고, Ansible 플레이북을 사용하여 자동으로 해결하는 데 도움이 되는 정보를 제공합니다. 고객 포털의 Knowledgebase 문서에 대한 링크도 제공됩니다.

A denial of service flaw exists in Samba that allows a remote attacker to supply crafted NetBIOS Session Service headers and cause an out-of-memory state. A remote attacker in a position to supply the crafted data can render system unusable.

[Knowledgebase article](#)

View the affected system

Total risk
The total risk of this remediation is **important**, based on the combination of likelihood and impact to remediate.

Critical likelihood
High impact

Risk of change
Very Low
The risk of change is **very low**, because the change takes very little time to implement and there is minimal impact to system operations.
System reboot is **not required**.

그림 1.10: 문제 세부 정보

어드바이저 서비스는 문제가 시스템에 제기하는 위험을 다음 두 가지 범주로 평가합니다.

총 위험

시스템에 문제가 미치는 영향을 나타냅니다.

변경 위험

시스템에 해결방안이 미치는 영향을 나타냅니다. 예를 들어 시스템을 재시작해야 할 수 있습니다.

취약점 서비스를 사용하여 보안 평가

취약점 서비스는 시스템에 영향을 주는 CVE(Common Vulnerabilities and Exposures)를 보고합니다. **Vulnerability > CVEs** 메뉴에서 서비스에 액세스합니다.

그림 1.11: 취약점 서비스의 보고서

Insights에서는 CVE마다에 추가 정보를 제공하고 노출된 시스템을 표시합니다. **Remediate** 버튼을 클릭하여 문제 해결을 위한 Ansible 플레이북을 생성할 수 있습니다.

그림 1.12: CVE의 세부 정보

규정 준수 서비스를 사용하여 규정 준수 분석

규정 준수 서비스는 시스템을 분석하고 규정 준수 수준을 OpenSCAP 정책에 보고합니다. OpenSCAP 프로젝트에서는 일련의 규칙에 대해 시스템의 규정 준수를 확인하는 도구를 구현합니다. Red Hat Insights는 PCI DSS(Payment Card Industry Data Security Standard)와 같은 다양한 정책을 기준으로 시스템을 평가하는 규칙을 제공합니다.

패치 서비스를 사용하여 패키지 업데이트

패치 서비스는 해당 시스템에 적용되는 Red Hat 제품 권고를 표시합니다. 적용 가능한 권고와 관련된 RPM 패키지를 업데이트하기 위해 실행할 수 있는 Ansible 플레이북도 생성할 수 있습니다. 특정 시스템의 권고 목록에 액세스하려면 **Patch > Systems** 메뉴를 사용합니다. **Apply all applicable advisories** 버튼을 클릭하여 시스템에 대한 Ansible 플레이북을 생성합니다.

그림 1.13: 시스템 패치

드리프트 서비스를 사용하여 시스템 비교

드리프트 서비스를 통해 시스템을 비교하거나 시스템 히스토리를 얻을 수 있습니다. 이 서비스를 사용하면 시스템을 유사 시스템이나 이전 시스템 상태와 비교하여 문제를 해결할 수 있습니다. **Drift > Comparison** 메뉴에서 서비스에 액세스할 수 있습니다.

다음 그림은 Red Hat Insights를 사용하여 동일한 시스템을 각기 다른 시간에 두 번 비교할 수 있음을 보여줍니다.

그림 1.14: 시스템 히스토리 비교

정책 서비스를 사용하여 경고 트리거

정책 서비스를 사용하면 시스템을 모니터링하는 규칙을 생성하고 시스템이 규칙을 준수하지 않을 경우 경고를 전송할 수 있습니다. Red Hat Insights는 시스템이 메타데이터를 동기화할 때마다 규칙을 평가합니다. **Policies** 메뉴에서 정책 서비스에 액세스할 수 있습니다.

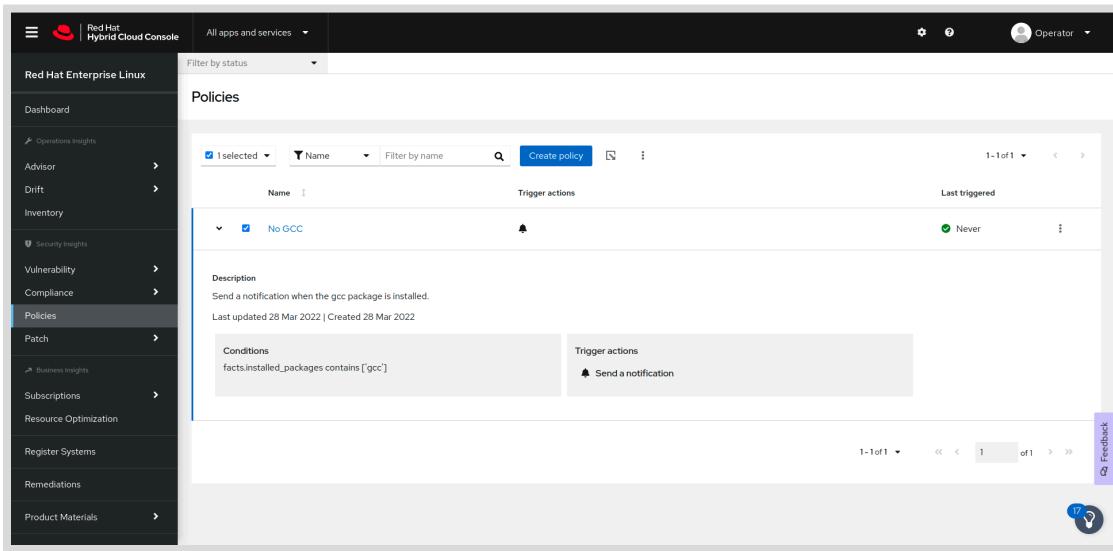


그림 1.15: 사용자 지정 규칙 세부 정보

인벤토리, 문제 해결 플레이북 및 서브스크립션 모니터링

Inventory 페이지에서는 Red Hat Insights에 등록한 시스템 목록을 제공합니다. **Last seen** 열에는 각 시스템에 대한 최신 메타데이터 업데이트 시간이 표시됩니다. 시스템 이름을 클릭하면 세부 정보를 검토하고 해당 시스템의 어드바이저, 취약점, 규정 준수 및 패치 서비스에 직접 액세스할 수 있습니다.

Remediations 페이지에는 수정을 위해 만든 모든 Ansible 플레이북이 나열되어 있습니다. 해당 페이지에서 플레이북을 다운로드할 수 있습니다.

Subscription 페이지를 사용하여 Red Hat 서브스크립션 사용량을 모니터링할 수 있습니다.

└
참조

insights-client(8) 및 **insights-client.conf(5)** 도움말 페이지

Red Hat Insights에 대한 자세한 내용은
https://access.redhat.com/documentation/en-us/red_hat_insights
 에서 Product Documentation for Red Hat Insights를 참조하십시오.

Insights에서 수집한 데이터를 제외하는 방법에 대한 자세한 내용은
https://access.redhat.com/documentation/en-us/red_hat_insights/2021/html-single/client_configuration_guide_for_red_hat_insights/assembly-main-client-cg
 에서 Client Configuration Guide for Red Hat Insights의 Red Hat Insights Client Data Obfuscation 및 Red Hat Insights Client Data Redaction 장을 참조하십시오.

Red Hat 인사이트에서 수집하는 데이터에 대한 정보는
Red Hat Insights에서 수집하는 시스템 정보
<https://access.redhat.com/articles/1598863>
 에서 확인할 수 있습니다.

▶ 퀴즈

Red Hat Insights를 사용하여 문제 탐지 및 해결

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. Red Hat Insights를 사용하여 Red Hat Enterprise Linux 시스템을 관리하는 경우 다음 이벤트는 어떤 순서로 발생합니까?

- 1) Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
 - 2) The Insights client uploads system metadata to the Red Hat Insights service.
 - 3) The administrator views the recommended actions in the Red Hat Insights customer portal.
 - 4) The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4
b. 4, 2, 1, 3
c. 4, 2, 3, 1
d. 4, 1, 2, 3

▶ 2. 다음 중 Red Hat Insights에 클라이언트를 등록하는 데 사용하는 명령은 무엇입니까?

- a. insights-client --register
- b. insights-client --no-upload
- c. subscription-manager register
- d. insights-client --unregister

▶ 3. 다음 중 시스템의 RPM 패키지를 업데이트하는 Ansible 플레이북을 생성할 수 있는 Red Hat Insights 콘솔 페이지는 무엇입니까?

- a. Advisor > Recommendations
- b. Vulnerability > Systems
- c. Patch > Systems
- d. Remediations

▶ 솔루션

Red Hat Insights를 사용하여 문제 탐지 및 해결

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. Red Hat Insights를 사용하여 Red Hat Enterprise Linux 시스템을 관리하는 경우 다음 이벤트는 어떤 순서로 발생합니까?

- 1) Red Hat Insights analyzes system metadata to determine which issues and recommendations apply.
 - 2) The Insights client uploads system metadata to the Red Hat Insights service.
 - 3) The administrator views the recommended actions in the Red Hat Insights customer portal.
 - 4) The Insights client collects system metadata on the Red Hat Enterprise Linux system.
- a. 1, 2, 3, 4
 - b. 4, 2, 1, 3
 - c. 4, 2, 3, 1
 - d. 4, 1, 2, 3

▶ 2. 다음 중 Red Hat Insights에 클라이언트를 등록하는 데 사용하는 명령은 무엇입니까?

- a. `insights-client --register`
- b. `insights-client --no-upload`
- c. `subscription-manager register`
- d. `insights-client --unregister`

▶ 3. 다음 중 시스템의 RPM 패키지를 업데이트하는 Ansible 플레이북을 생성할 수 있는 Red Hat Insights 콘솔 페이지는 무엇입니까?

- a. Advisor > Recommendations
- b. Vulnerability > Systems
- c. Patch > Systems
- d. Remediations

요약

- 하나 이상의 명령줄 텍스트 편집기를 사용하는 방법을 알아야 합니다. 일반적으로 Linux 배포판에 기본적으로 설치되는 Vim을 사용하는 것이 좋습니다.
- SSH는 암호 기반 인증과 키 기반 인증을 모두 지원합니다.
- **ssh-keygen** 명령은 인증을 위해 SSH 키 쌍을 생성합니다. **ssh-copy-id** 명령은 공개 키를 원격 시스템으로 내보냅니다.
- Red Hat Customer Portal에서는 설명서, 다운로드, 최적화 도구, 지원 사례 관리, Red Hat 제품의 서비스 크립션 및 자격 관리에 액세스할 수 있습니다.
- Red Hat Insights는 시스템의 보안, 성능, 가용성 및 안정성에 대한 위협을 식별하고 해결하는 데 도움이 되는 SaaS 기반 예측 분석 툴입니다.

2장

명령줄에서 파일 관리

목적

Bash 쉘에서 파일을 복사, 이동, 생성, 삭제 및 구성합니다.

목표

- Linux에서 파일을 구성하는 방법 및 파일 시스템 계층 구조에 있는 여러 디렉터리의 용도를 설명합니다.
- 하드 링크와 심볼릭(또는 "소프트") 링크를 사용하여 여러 파일 이름이 동일한 파일을 참조하도록 합니다.
- Bash 쉘의 패턴 일치 기능을 사용하여 많은 파일에 영향을 주는 명령을 효율적으로 실행합니다.

섹션

- Linux 파일 시스템 계층 구조 개념 설명(퀴즈)
- 파일 간 링크 만들기(안내에 따른 연습)
- 쉘 확장을 사용하여 파일 이름 일치(퀴즈)

랩

- 명령줄에서 파일 관리

Linux 파일 시스템 계층 구조 개념 설명

목표

Linux에서 파일을 구성하는 방법 및 파일 시스템 계층 구조에 있는 여러 디렉터리의 용도를 설명합니다.

파일 시스템 계층 구조

Linux 시스템은 파일 시스템 계층 구조라는 반전된 단일 트리로 구성된 파일 시스템에 모든 파일을 저장합니다. 이 계층 구조는 트리 루트가 맨 위에 있고 디렉터리와 하위 디렉터리의 분기가 루트 아래로 뻗어 가기 때문에 반전된 트리입니다.

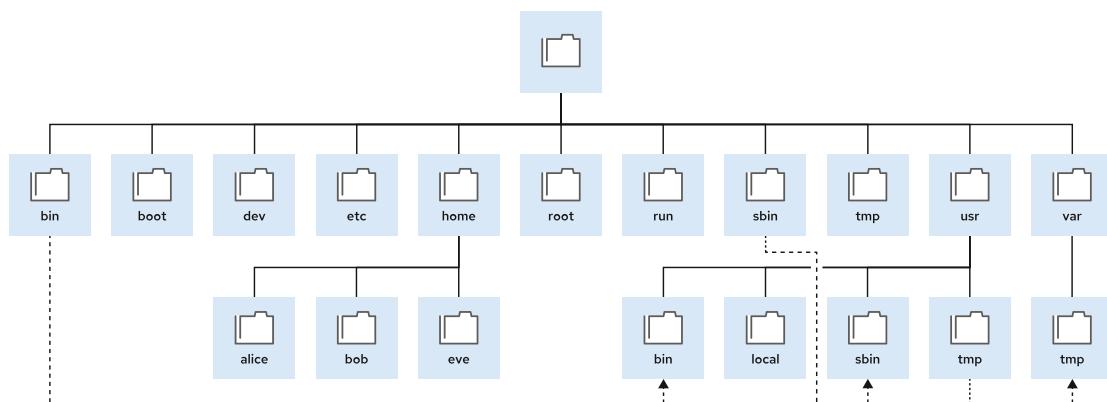


그림 2.1: Red Hat Enterprise Linux 9에서 중요한 파일 시스템 디렉터리

/ 디렉터리는 파일 시스템 계층 구조의 최상단에 있는 루트 디렉터리입니다. / 문자는 파일 이름에서 디렉터리 구분자로도 사용됩니다. 예를 들어 `etc` 가 / 디렉터리의 하위 디렉터리인 경우 해당 디렉터리를 `/etc`로 나타냅니다 마찬가지로, `/etc` 디렉터리가 `issue` 파일을 포함하는 경우 해당 파일을 `/etc/issue`로 나타냅니다.

/ 의 하위 디렉터리는 파일을 찾기 쉽도록 유형 및 목적에 따라 파일을 구성하기 위한 표준화된 목적에 사용됩니다. 예를 들어 루트 디렉터리에서 `/boot` 하위 디렉터리는 시스템을 부팅하기 위한 파일을 저장하는데 사용됩니다.



참고

다음 용어는 파일 시스템 디렉터리 내용을 설명하는데 도움이 됩니다.

- 정적 내용은 명시적으로 편집 또는 재구성될 때까지 변경되지 않습니다.
- 동적 또는 가변 내용은 활성 프로세스에서 수정 또는 추가될 수 있습니다.
- 영구적 내용은 구성 설정과 같이 재부팅 후에도 유지됩니다.
- 프로세스 또는 시스템의 런타임 내용은 재부팅 시 삭제됩니다.

다음 표에는 시스템의 몇 가지 중요한 디렉터리가 이름과 목적별로 나와 있습니다.

중요한 Red Hat Enterprise Linux 디렉터리

위치	목적
/boot	부팅 프로세스를 시작하기 위한 파일입니다.
/dev	시스템에서 하드웨어에 액세스하는 데 사용하는 특수 장치 파일입니다.
/etc	시스템별 구성 파일입니다.
/home	일반 사용자가 데이터와 구성 파일을 저장하는 홈 디렉터리입니다.
/root	관리자 수퍼유저의 홈 디렉터리는 root입니다.
/run	마지막 부팅 이후 시작된 프로세스의 런타임 데이터입니다. 이 데이터에는 프로세스 ID 파일과 잠금 파일이 포함됩니다. 이 디렉터리의 내용은 재부팅 시 다시 생성됩니다. 이 디렉터리는 이전 버전의 Red Hat Enterprise Linux에 있던 /var/run 및 /var/lock 디렉터리를 통합합니다.
/tmp	어디에서나 쓸 수 있는 임시 파일용 공간입니다. 10일 동안 액세스, 변경 또는 수정되지 않은 파일은 이 디렉터리에서 자동으로 삭제됩니다. /var/tmp 디렉터리도 임시 디렉터리로, 30일 이상 액세스, 변경 또는 수정되지 않은 파일은 자동으로 삭제됩니다.
/usr	설치된 소프트웨어, 공유 라이브러리, 포함 파일 및 읽기 전용 프로그램 데이터입니다. /usr 디렉터리의 중요한 하위 디렉터리에는 다음 명령이 포함됩니다. <ul style="list-style-type: none"> • /usr/bin: 사용자 명령 • /usr/sbin: 시스템 관리 명령 • /usr/local: 로컬 사용자 지정 소프트웨어
/var	재부팅 후에도 유지되는 시스템별 가변 데이터입니다. 동적으로 변경되는 파일(예: 데이터베이스, 캐시 디렉터리, 로그 파일, 프린터로 전송된 문서, 웹 사이트 콘텐츠)은 /var에서 찾을 수 있습니다.



중요

Red Hat Enterprise Linux 7 이상에서 /의 오래된 디렉터리 4개에는 /usr의 상응하는 디렉터리와 동일한 내용이 포함됩니다.

- /bin 계약은 /usr/bin
- /sbin 계약은 /usr/sbin
- /lib 계약은 /usr/lib
- /lib64 계약은 /usr/lib64

이전 버전의 Red Hat Enterprise Linux에서는 서로 다른 파일 세트를 포함하는 별도의 디렉터리였습니다. Red Hat Enterprise Linux 7 이상에서 /의 디렉터리는 /usr의 일치하는 디렉터리에 대한 심볼릭 링크입니다.



참조

[hier\(7\)](#) 도움말 페이지

▶ 퀴즈

Linux 파일 시스템 계층 구조 개념 설명

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 영구적인 시스템 특정 구성 데이터가 포함되어 있는 디렉터리는 무엇입니까?

- a. /etc
- b. /root
- c. /run
- d. /usr

▶ 2. 다음 중 시스템의 파일 시스템 계층 구조의 최상위 디렉터리는 무엇입니까?

- a. /etc
- b. /
- c. /home/root
- d. /root

▶ 3. 다음 중 사용자 홈 디렉터리가 들어 있는 디렉터리는 무엇입니까?

- a. /
- b. /home
- c. /root
- d. /user

▶ 4. 다음 중 시스템을 부팅하기 위한 파일이 포함된 디렉터리는 무엇입니까?

- a. /boot
- b. /home/root
- c. /bootable
- d. /etc

▶ 5. 다음 중 하드웨어에 액세스하기 위한 시스템 파일이 포함된 디렉터리는 무엇입니까?

- a. /etc
- b. /run
- c. /dev
- d. /usr

▶ 6. 다음 중 관리자 수퍼유저의 홈 디렉터리는 무엇입니까?

- a. /etc
- b. /
- c. /home/root
- d. /root

▶ 7. 다음 중 일반 명령과 유ти리티가 들어 있는 디렉터리는 무엇입니까?

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

▶ 8. 다음 중 비영구적인 프로세스 런타임 데이터가 포함되어 있는 디렉터리는 무엇입니까?

- a. /tmp
- b. /etc
- c. /run
- d. /var

▶ 9. 다음 중 설치된 소프트웨어 프로그램 및 라이브러리가 들어 있는 디렉터리는 무엇입니까?

- a. /etc
- b. /lib
- c. /usr
- d. /var

▶ 솔루션

Linux 파일 시스템 계층 구조 개념 설명

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 영구적인 시스템 특정 구성 데이터가 포함되어 있는 디렉터리는 무엇입니까?

- a. /etc
- b. /root
- c. /run
- d. /usr

▶ 2. 다음 중 시스템의 파일 시스템 계층 구조의 최상위 디렉터리는 무엇입니까?

- a. /etc
- b. /
- c. /home/root
- d. /root

▶ 3. 다음 중 사용자 홈 디렉터리가 들어 있는 디렉터리는 무엇입니까?

- a. /
- b. /home
- c. /root
- d. /user

▶ 4. 다음 중 시스템을 부팅하기 위한 파일이 포함된 디렉터리는 무엇입니까?

- a. /boot
- b. /home/root
- c. /bootable
- d. /etc

▶ 5. 다음 중 하드웨어에 액세스하기 위한 시스템 파일이 포함된 디렉터리는 무엇입니까?

- a. /etc
- b. /run
- c. /dev
- d. /usr

▶ 6. 다음 중 관리자 수퍼유저의 홈 디렉터리는 무엇입니까?

- a. /etc
- b. /
- c. /home/root
- d. /root

▶ 7. 다음 중 일반 명령과 유ти리티가 들어 있는 디렉터리는 무엇입니까?

- a. /commands
- b. /run
- c. /usr/bin
- d. /usr/sbin

▶ 8. 다음 중 비영구적인 프로세스 런타임 데이터가 포함되어 있는 디렉터리는 무엇입니까?

- a. /tmp
- b. /etc
- c. /run
- d. /var

▶ 9. 다음 중 설치된 소프트웨어 프로그램 및 라이브러리가 들어 있는 디렉터리는 무엇입니까?

- a. /etc
- b. /lib
- c. /usr
- d. /var

파일 간 링크 만들기

목표

하드 링크와 심볼릭(또는 "소프트") 링크를 사용하여 여러 파일 이름이 동일한 파일을 참조하도록 합니다.

파일 간 링크 관리

동일한 파일을 가리키는 여러 개의 파일 이름을 생성할 수 있습니다. 이러한 파일 이름을 링크라고 합니다.

하드 링크 또는 심볼릭 링크(소프트 링크라고도 함)라는 두 가지 유형의 링크를 생성할 수 있습니다. 각 방법에는 장단점이 있습니다.

하드 링크 생성

모든 파일은 초기 이름부터 파일 시스템의 데이터까지 단일 하드 링크로 시작합니다. 파일에 대한 하드 링크를 생성하는 경우 동일한 데이터를 가리키는 다른 이름을 생성합니다. 새로운 하드 링크는 원본 파일 이름과 동일하게 작동합니다. 링크가 생성된 후에는 새로운 하드 링크와 원래 파일 이름을 구분할 수 없습니다.

`ls -l` 명령을 사용하여 파일에 여러 개의 하드 링크가 있는지 확인할 수 있습니다. 보고되는 항목 중 하나는 파일이 보유한 하드 링크 수인 각 파일의 링크 수입니다. 다음 예제에서 `newfile.txt` 파일의 링크 수는 1입니다. `/home/user/newfile.txt` 위치인 정확히 1개의 절대 경로가 있습니다.

```
[user@host ~]$ pwd
/home/user
[user@host ~]$ ls -l newfile.txt
-rw-r--r--. 1 user user 0 Mar 11 19:19 newfile.txt
```

`ln` 명령을 사용하여 기존 파일을 가리키는 하드 링크(다른 파일 이름)를 생성할 수 있습니다. 이 명령에는 최소한 두 개의 인수인 기존 파일의 경로 및 생성하려는 하드 링크의 경로가 필요합니다.

다음 예제에서는 `/tmp` 디렉터리에 있는 기존 `newfile.txt` 파일에 대해 `newfile-hlink2.txt`라는 하드 링크를 생성합니다.

```
[user@host ~]$ ln newfile.txt /tmp/newfile-hlink2.txt
[user@host ~]$ ls -l newfile.txt /tmp/newfile-hlink2.txt
-rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt
-rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

두 파일이 하드 링크되었는지 확인하려면 `ls` 명령의 `-i` 옵션을 사용하여 각 파일의 inode 번호를 표시합니다. 파일이 동일한 파일 시스템에 있고 inode 번호가 같으면 동일한 데이터 파일 내용을 가리키는 하드 링크입니다.

```
[user@host ~]$ ls -il newfile.txt /tmp/newfile-hlink2.txt
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 newfile.txt
8924107 -rw-rw-r--. 2 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
```

**중요**

동일한 파일을 참조하는 하드 링크는 링크 수, 액세스 권한, 사용자 및 그룹 소유권, 타임스탬프 및 파일 내용으로 이루어진 inode 구조를 공유합니다. 하나의 하드 링크에 대한 정보가 변경되면 동일한 파일의 다른 하드 링크에도 새 정보가 표시됩니다. 이런 동작은 각 하드 링크가 스토리지의 동일한 데이터를 가리키기 때문입니다.

원본 파일이 삭제된 경우에도 적어도 하나의 다른 하드 링크가 있는 한, 파일 내용에 액세스할 수 있습니다. 마지막 하드 링크가 삭제되어 파일 내용을 참조하는 하드 링크가 없는 경우에만 스토리지에서 데이터가 삭제됩니다.

```
[user@host ~]$ rm -f newfile.txt
[user@host ~]$ ls -l /tmp/newfile-hlink2.txt
-rw-rw-r-- 1 user user 12 Mar 11 19:19 /tmp/newfile-hlink2.txt
[user@host ~]$ cat /tmp/newfile-hlink2.txt
Hello World
```

하드 링크의 한계

하드 링크에는 몇 가지 한계가 있습니다. 첫째, 하드 링크는 일반 파일에서만 사용할 수 있습니다. `ln` 명령을 사용하여 디렉터리 또는 특수 파일에 대한 하드 링크를 생성할 수 없습니다.

둘째, 하드 링크는 두 파일이 동일한 파일 시스템에 있는 경우에만 사용할 수 있습니다. 파일 시스템 계층 구조는 여러 개의 스토리지 장치로 구성될 수 있습니다. 시스템 구성에 따라 새 디렉터리로 변경할 때 해당 디렉터리와 내용이 다른 파일 시스템에 저장될 수도 있습니다.

`df` 명령을 사용하여 다른 파일 시스템에 있는 디렉터리를 표시할 수 있습니다. 예를 들어 다음 출력이 표시될 수 있습니다.

```
[user@host ~]$ df
Filesystem      1K-blocks   Used Available Use% Mounted on
devtmpfs          886788     0  886788  0% /dev
tmpfs             902108     0  902108  0% /dev/shm
tmpfs             902108   8696  893412  1% /run
tmpfs             902108     0  902108  0% /sys/fs/cgroup
/dev/mapper/rhel_rhel9--root 10258432 1630460  8627972 16% /
/dev/sda1        1038336  167128  871208 17% /boot
tmpfs            180420     0  180420  0% /run/user/1000
```

서로 다른 두 개의 "마운트된" 디렉터리에 있는 파일과 하위 디렉터리는 다른 파일 시스템에 있습니다. 따라서 이 예제의 시스템에서는 `/var/tmp/link1` 및 `/home/user/file` 파일 간에 하드 링크를 생성할 수 있습니다. 둘 다 /디렉터리의 하위 디렉터리지만 목록에 있는 다른 디렉터리의 하위 디렉터리가 아니기 때문입니다. 그러나 `/boot/test/badlink` 및 `/home/user/file` 파일 간에 하드 링크를 만들 수 없습니다. 첫 번째 파일은 `/boot` 디렉터리의 하위 디렉터리("Mounted on" 목록에 있음)에 있으며 `/dev/sda1` 파일 시스템에 있습니다. 두 번째 파일은 `/dev/mapper/rhel_rhel9--root` 파일 시스템에 있습니다.

심볼릭 링크 생성

`ln` 명령의 `-s` 옵션은 "소프트 링크"라고도 하는 심볼릭 링크를 생성합니다. 심볼릭 링크는 일반 파일이 아니라 기존 파일 또는 디렉터리를 가리키는 특별한 유형의 파일입니다.

심볼릭 링크는 하드 링크 대비 몇 가지 장점이 있습니다.

2장 | 명령줄에서 파일 관리

- 심볼릭 링크는 서로 다른 파일 시스템에 있는 두 개의 파일을 연결할 수 있습니다.
- 심볼릭 링크는 일반 파일뿐 아니라 디렉터리나 특수 파일을 가리킬 수 있습니다.

다음 예제에서 `ln -s` 명령은 `/home/user/newfile-link2.txt` 파일에 대한 심볼릭 링크를 생성합니다. 심볼릭 링크의 이름은 `/tmp/newfile-symlink.txt`입니다.

```
[user@host ~]$ ln -s /home/user/newfile-link2.txt /tmp/newfile-symlink.txt
[user@host ~]$ ls -l newfile-link2.txt /tmp/newfile-symlink.txt
-rw-rw-r--. 1 user user 12 Mar 11 19:19 newfile-link2.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
Symbolic Hello World
```

앞의 예제에서 `/tmp/newfile-symlink.txt` 파일에 대한 긴 목록의 첫 번째 문자는 `-`가 아닌 `l`(문자 l)입니다. 이 문자는 파일이 일반 파일이 아닌 심볼릭 링크임을 나타냅니다.

원본 일반 파일이 삭제된 경우 심볼릭 링크는 계속해서 해당 파일을 가리키지만 타겟이 없습니다. 누락된 파일을 가리키는 심볼릭 링크를 "매달린 심볼릭 링크"라고 합니다.

```
[user@host ~]$ rm -f newfile-link2.txt
[user@host ~]$ ls -l /tmp/newfile-symlink.txt
lrwxrwxrwx. 1 user user 11 Mar 11 20:59 /tmp/newfile-symlink.txt -> /home/user/
newfile-link2.txt
[user@host ~]$ cat /tmp/newfile-symlink.txt
cat: /tmp/newfile-symlink.txt: No such file or directory
```



중요

앞의 예제에서 매달린 심볼릭 링크의 부작용 중 하나는 삭제된 파일(`/home/user/newfile-link2.txt`)과 동일한 이름의 파일을 생성하는 경우 심볼릭 링크는 더 이상 "매달린" 상태가 아니며 새 파일을 가리킨다는 것입니다. 하드 링크는 이런 방식으로 작동하지 않습니다. 하드 링크를 삭제한 다음 `ln` 대신 일반 툴을 사용하여 같은 이름의 파일을 생성하는 경우에는 새 파일이 이전 파일에 연결되지 않습니다. 하드 링크와 심볼릭 링크를 다음과 같이 비교하여 작동 방식을 파악하십시오.

- 하드 링크에서는 이름이 스토리지 장치의 데이터를 가리킵니다.
- 심볼릭 링크에서는 이름이 스토리지 장치의 데이터를 가리키는 다른 이름을 가리킵니다.

심볼릭 링크는 디렉터리를 가리킬 수 있습니다. 이 경우 심볼릭 링크는 디렉터리와 같은 역할을 합니다. `cd`를 사용하여 심볼릭 링크로 변경하는 경우 현재 작업 디렉터리는 연결된 디렉터리가 됩니다. 일부 툴은 심볼릭 링크를 따라 도달했다는 것을 추적할 수 있습니다. 예를 들어 `cd`는 기본적으로 실제 디렉터리의 이름 대신 심볼릭 링크의 이름을 사용하여 현재 작업 디렉터리를 업데이트합니다. 실제 디렉터리의 이름을 사용하여 현재 작업 디렉터리를 업데이트하려는 경우 `-P` 옵션을 사용할 수 있습니다.

다음 예제에서 `/etc` 디렉터리를 가리키는 `/home/user/configfiles`라는 심볼릭 링크를 생성됩니다.

```
[user@host ~]$ ln -s /etc /home/user/configfiles
[user@host ~]$ cd /home/user/configfiles
[user@host configfiles]$ pwd
/home/user/configfiles
[user@host configfiles]$ cd -P /home/user/configfiles
[user@host etc]$ pwd
/etc
```



참조

[ln\(1\) 도움말 페이지](#)

info ln(Make links between files)

▶ 연습 가이드

파일 간 링크 만들기

이 연습에서는 하드 링크와 심볼릭 링크를 생성하고 결과를 비교합니다.

결과

- 파일 간 하드 링크와 심볼릭 링크를 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start files-make
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다. 시스템 구성에 따라 인증을 위해 SSH 키를 사용할 수 있으므로 암호는 필요하지 않습니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **/home/student/files/target.file** 파일에 대한 **/home/student/links/file.hardlink**라는 하드 링크를 생성합니다. 원본 파일과 새로 연결된 파일의 링크 수를 확인합니다.

2.1. **/home/student/files/target.file** 파일의 링크 수를 봅니다.

```
[student@servera ~]$ ls -l files/target.file
total 4
-rw-r--r--. 1 student student 11 Mar  3 06:51 files/target.file
```

2.2. **/home/student/links/file.hardlink**라는 하드 링크를 생성합니다. 이 하드 링크를 **/home/student/files/target.file** 파일에 연결합니다.

```
[student@servera ~]$ ln /home/student/files/target.file \
/home/student/links/file.hardlink
```

2.3. 원본 **/home/student/files/target.file** 파일과 새로 연결된 파일 **/home/student/files/file.hardlink**의 링크 수를 확인합니다. 두 파일의 링크 수가 모두 2여야 합니다.

```
[student@servera ~]$ ls -l files/target.file links/file.hardlink
-rw-r--r-- 2 student student 11 Mar 3 06:51 files/target.file
-rw-r--r-- 2 student student 11 Mar 3 06:51 links/file.hardlink
```

- ▶ 3. servera 시스템의 /tmp 디렉터리를 가리키는 /home/student/tempdir이라는 심볼릭 링크를 생성됩니다. 새로 생성된 심볼릭 링크를 확인합니다.

3.1. /home/student/tempdir이라는 심볼릭 링크를 생성하여 /tmp 디렉터리에 연결합니다.

```
[student@servera ~]$ ln -s /tmp /home/student/tempdir
```

3.2. ls -l 명령을 사용하여 새로 생성한 심볼릭 링크를 확인합니다.

```
[student@servera ~]$ ls -l /home/student/tempdir
lrwxrwxrwx. 1 student student 4 Mar 3 06:55 /home/student/tempdir -> /tmp
```

- ▶ 4. workstation 시스템에 student 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish files-make
```

이것으로 섹션을 완료합니다.

쉘 확장을 사용하여 파일 이름 일치

목표

Bash 쉘의 패턴 일치 기능을 사용하여 많은 파일에 영향을 주는 명령을 효율적으로 실행합니다.

명령줄 확장

Bash 쉘 프롬프트에서 명령을 입력하면 쉘은 명령줄을 실행하기 전에 여러 번의 확장을 통해 해당 명령줄을 처리합니다. 이러한 쉘 확장을 사용하여 어렵거나 불가능했던 복잡한 작업을 수행할 수 있습니다.

Bash 쉘에서 수행하는 주요 확장은 다음과 같습니다.

- 여러 문자열을 생성할 수 있는 중괄호 확장
- 사용자 홈 디렉터리 경로로 확장되는 틸드 확장
- 텍스트를 쉘 변수에 저장된 값으로 바꾸는 변수 확장
- 텍스트를 명령 출력으로 바꾸는 명령 대체
- 패턴 일치를 통해 하나 이상의 파일을 선택하는 데 도움을 주는 경로 이름 확장

글러빙(globbing)이라고 하는 경로 이름 확장은 Bash의 가장 유용한 기능 중 하나입니다. 이 기능을 사용하면 많은 파일을 더 쉽게 관리할 수 있습니다. 검색 중인 파일 및 경로 이름과 일치하도록 '확장'되는 메타 문자를 사용하면 특정 파일 집합에 대해 명령을 한 번에 실행할 수 있습니다.

경로 이름 확장 및 패턴 일치

경로 이름 확장은 와일드카드 또는 문자 클래스를 나타내는 특수 문자 패턴을 해당 패턴과 일치하는 파일 이름 목록으로 확장합니다. 쉘은 명령을 실행하기 전에 패턴을 일치하는 파일 이름 목록으로 바꿉니다. 패턴이 어느 것과도 일치하지 않는 경우 쉘은 패턴을 실행하는 명령의 리터럴 인수로 사용하려고 합니다. 다음 표에는 패턴 일치에 사용되는 일반적인 메타 문자와 패턴 클래스가 나와 있습니다.

메타 문자 및 일치 테이블

패턴	일치
*	0개 이상의 문자로 이루어진 모든 문자열
?	모든 단일 문자
[abc···]	대괄호에 포함된 한 문자
[!abc···]	대괄호에 포함되지 않은 한 문자
[^abc···]	대괄호에 포함되지 않은 한 문자
[[:alpha:]]	알파벳 문자
[[:lower:]]	소문자
[[:upper:]]	대문자

패턴	일치
<code>[:alnum:]</code>	알파벳 문자 또는 숫자
<code>[:punct:]</code>	공백 또는 영숫자가 아닌 출력 가능한 문자
<code>[:digit:]</code>	0에서 9 사이의 한 자리 숫자
<code>[:space:]</code>	탭, 줄바꿈, 캐리지 리턴, 용지 공급 또는 공백을 포함할 수 있는 단일 공백 문자

다음 예제를 위해 다음 명령을 실행하여 몇 개의 샘플 파일을 생성했다고 가정합니다.

```
[user@host ~]$ mkdir glob; cd glob
[user@host glob]$ touch alfa bravo charlie delta echo able baker cast dog easy
[user@host glob]$ ls
able alfa baker bravo cast charlie delta dog easy echo
[user@host glob]$
```

다음 예제에서 처음 두 개의 명령은 별표(*)가 있는 간단한 패턴 일치를 사용하여 각각 "a"로 시작하는 모든 파일 이름과 "a"가 포함된 모든 파일 이름을 찾습니다. 세 번째 명령은 별표와 대괄호를 사용하여 "a" 또는 "c"로 시작하는 모든 파일 이름을 찾습니다.

```
[user@host glob]$ ls a*
able alfa
[user@host glob]$ ls *a*
able alfa baker bravo cast charlie delta easy
[user@host glob]$ ls [ac]*
able alfa cast charlie
```

또한 다음 예제에서는 물음표(?) 문자를 사용하여 해당 파일 이름 중 일부를 찾습니다. 두 명령은 각각 길이가 4자와 5자인 파일 이름만 찾습니다.

```
[user@host glob]$ ls ****
able cast easy echo
[user@host glob]$ ls ??????
baker bravo delta
```

중괄호 확장

중괄호 확장은 임의의 문자열을 생성하는데 사용됩니다. 중괄호에는 쉼표로 구분된 문자열 목록이나 시퀀스식이 포함됩니다. 결과에는 중괄호 정의의 앞이나 뒤에 오는 텍스트가 포함됩니다. 중괄호 확장은 하나가 다른 하나의 안쪽에 중첩될 수 있습니다. 시퀀스로 확장되는 이중 점 구문(..)을 사용할 수도 있습니다. 예를 들어 중괄호 안의 `{m..p}` 이중 점 구문은 `m n o p`로 확장됩니다.

```
[user@host glob]$ echo {Sunday,Monday,Tuesday,Wednesday}.log
Sunday.log Monday.log Tuesday.log Wednesday.log
[user@host glob]$ echo file{1..3}.txt
file1.txt file2.txt file3.txt
[user@host glob]$ echo file{a..c}.txt
filea.txt fileb.txt filec.txt
[user@host glob]$ echo file{a,b}{1,2}.txt
```

2장 | 명령줄에서 파일 관리

```
filea1.txt filea2.txt fileb1.txt fileb2.txt
[user@host glob]$ echo file{a[1,2],b,c}.txt
filea1.txt filea2.txt fileb.txt filec.txt
```

중괄호 확장을 실제로 사용하면 여러 개의 파일이나 디렉터리를 생성할 수 있습니다.

```
[user@host glob]$ mkdir ..../RHEL{7,8,9}
[user@host glob]$ ls ..../RHEL*
RHEL7 RHEL8 RHEL9
```

틸드 확장

틸드 문자(~)는 현재 사용자의 홈 디렉터리와 일치합니다. 슬래시(/) 이외의 문자열로 시작하면 쉘은 해당 슬래시까지의 문자열을 사용자 이름으로 해석하고(일치 항목이 있는 경우), 사용자 홈 디렉터리의 절대 경로로 문자열을 대체합니다. 일치하는 사용자 이름이 없으면 쉘은 차례로 실제 틸드와 문자열을 함께 사용합니다.

다음 예제에서 **echo** 명령은 틸드 문자의 값을 표시하는 데 사용됩니다. **echo** 명령을 사용하여 중괄호 및 변수 확장 문자 등의 값을 표시할 수도 있습니다.

```
[user@host glob]$ echo ~root
/root
[user@host glob]$ echo ~user
/home/user
[user@host glob]$ echo ~/glob
/home/user/glob
[user@host glob]$ echo ~nonexistinguser
~nonexistinguser
```

변수 확장

변수는 메모리에 값을 저장하는 명명된 컨테이너와 같은 역할을 합니다. 변수를 사용하면 명령줄이나 쉘 스크립트 내에서 저장된 데이터에 간단하게 액세스하고 수정할 수 있습니다.

다음 구문을 사용하여 변수에 값으로 데이터를 할당할 수 있습니다.

```
[user@host ~]$ VARIABLENAME=value
```

변수 확장을 사용하여 변수 이름을 명령줄의 값으로 변환할 수 있습니다. 문자열이 달려 기호(\$)로 시작하는 경우 쉘은 해당 문자열의 나머지를 변수 이름으로 사용하고 변수 값으로 대체합니다.

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo $USERNAME
operator
```

다른 쉘 확장으로 인한 실수를 방지하기 위해 변수 이름을 중괄호로 묶을 수 있습니다(예: \${VARIABLENAME}).

```
[user@host ~]$ USERNAME=operator
[user@host ~]$ echo ${USERNAME}
operator
```

변수 이름에는 문자(대문자 및 소문자), 숫자, 밑줄만 포함할 수 있습니다. 변수 이름은 대소문자를 구분하며 숫자로 시작할 수 없습니다.

명령 대체

명령 대체를 사용하면 명령 출력으로 명령줄에서 명령 자체를 대체할 수 있습니다. 명령이 괄호로 묶여 있고 달러 기호(\$)가 앞에 오면 명령 대체가 발생합니다. **\$(command)** 형식은 서로의 안쪽에 복수의 명령 확장을 중첩시킬 수 있습니다.

```
[user@host glob]$ echo Today is $(date +%A).
Today is Wednesday.
[user@host glob]$ echo The time is $(date +%M) minutes past $(date +%l%p).
The time is 26 minutes past 11AM.
```



참고

이전 형식의 명령 대체에서는 `command`과 같이 백틱을 사용합니다. Bash 쉘은 이 형식을 계속 허용하지만, 시각적으로 백틱을 작은따옴표와 혼동하기 쉽고 백틱을 중첩할 수 없으므로 사용하지 않는 것이 좋습니다.

인수 확장 방지

Bash 쉘에서는 많은 문자가 특수한 의미를 가집니다. 명령줄 일부에서 쉘 확장을 방지하기 위해 문자 및 문자열을 따옴표로 묶고 이스케이프 처리할 수 있습니다.

백슬래시(\)는 Bash 쉘에서 이스케이프 문자입니다. 다음 문자가 확장되지 않도록 보호합니다.

```
[user@host glob]$ echo The value of $HOME is your home directory.
The value of /home/user is your home directory.
[user@host glob]$ echo The value of \$HOME is your home directory.
The value of $HOME is your home directory.
```

앞의 예제에서는 달러 기호가 확장되지 않도록 보호되며, Bash에서 **\$HOME**의 변수 확장 없이 일반 문자로 취급됩니다.

더 긴 문자열을 보호하기 위해 작은따옴표(') 또는 큰따옴표(")를 사용하여 문자열을 묶을 수 있습니다. 약간 다른 효과가 있습니다. 작은따옴표는 모든 쉘 확장을 중지합니다. 큰따옴표는 대부분의 쉘 확장을 중지합니다.

큰따옴표를 사용하면 달러 기호(\$), 백슬래시(\), 백틱(`), 느낌표(!) 이외의 특수 문자가 따옴표로 묶인 텍스트 내에서 작동하지 않습니다. 큰따옴표는 경로 이름을 확장하지 않지만 명령 대체 및 변수 확장은 계속 허용합니다.

```
[user@host glob]$ myhost=$(hostname -s); echo $myhost
host
[user@host glob]$ echo "***** hostname is ${myhost} *****"
***** hostname is host *****
```

작은따옴표를 사용하면 따옴표로 묶인 모든 텍스트가 문자 그대로 해석됩니다.

```
[user@host glob]$ echo "Will variable $myhost evaluate to $(hostname -s)?"
Will variable host evaluate to host?
[user@host glob]$ echo 'Will variable $myhost evaluate to $(hostname -s)?'
Will variable $myhost evaluate to $(hostname -s)?
```



중요

화면과 키보드 둘 다에서 작은따옴표(')와 명령 대체 백틱(`)을 혼동하기 쉽습니다. 잘못 사용할 경우 예기치 않은 쉘 동작이 발생합니다.



참조

`bash(1)`, `cd(1)`, `glob(7)`, `isalpha(3)`, `ls(1)`, `path_resolution(7)` 및 `pwd(1)` 도움말 페이지

▶ 퀴즈

쉘 확장을 사용하여 파일 이름 일치

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 "b"로 끝나는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

▶ 2. 다음 중 "b"로 시작하는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

▶ 3. 다음 중 첫 번째 문자가 "b"가 아닌 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

▶ 4. 다음 중 "b"를 포함하는 모든 파일 이름과 일치하는 패턴은 무엇입니까?

- a. **b***
- b. ***b**
- c. ***b***
- d. **[!b]***

▶ 5. 다음 중 숫자를 포함하는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. ***#***
- b. ***[[digit:]]***
- c. ***[digit]***
- d. **[0-9]**

▶ 6. 다음 중 대문자로 시작되는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. ^?*
- b. ^*
- c. [upper]*
- d. [:upper:]*
e. [[CAP]]*

▶ 7. 다음 중 3자 이상인 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. ???*
- b. ???
- c. \3*
- d. +++*
- e. ... *

▶ 솔루션

쉘 확장을 사용하여 파일 이름 일치

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 "b"로 끝나는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

▶ 2. 다음 중 "b"로 시작하는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

▶ 3. 다음 중 첫 번째 문자가 "b"가 아닌 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

▶ 4. 다음 중 "b"를 포함하는 모든 파일 이름과 일치하는 패턴은 무엇입니까?

- a. b*
- b. *b
- c. *b*
- d. [!b]*

▶ 5. 다음 중 숫자를 포함하는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. *#*
- b. *[[:digit:]]*
- c. *[digit]*
- d. [0-9]

▶ 6. 다음 중 대문자로 시작되는 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. ^?*
- b. ^*
- c. [upper]*
- d. [[:upper:]]*
- e. [[CAP]]*

▶ 7. 다음 중 3자 이상인 파일 이름하고만 일치하는 패턴은 무엇입니까?

- a. ???*
- b. ???
- c. \3*
- d. +++*
- e. ... *

▶ 랩

명령줄에서 파일 관리

이 랩에서는 쉘과 다양한 파일 이름 일치 기술을 사용하여 파일과 디렉터리를 효율적으로 생성, 이동 및 제거합니다.

결과

- 와일드카드를 사용하여 파일을 찾아서 조작합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start files-review
```

지침

1. **ssh** 명령을 사용하여 **student** 사용자로 **serverb** 시스템에 로그인합니다. 시스템 구성에 따라 인증을 위해 SSH 키를 사용할 수 있습니다.
2. **Documents** 디렉터리에 **project_plans**라는 디렉터리를 생성합니다. **Documents** 디렉터리는 **student** 사용자의 홈 디렉터리에 있어야 합니다. **project_plans** 디렉터리에 **season1_project_plan.odf** 및 **season2_project_plan.odf**라는 빈 파일 2개를 생성합니다.
힌트: **~/Documents** 디렉터리가 없는 경우 **mkdir** 명령의 **-p** 옵션을 사용하여 생성합니다.
3. 이 랩에서 사용할 빈 연습 파일 세트를 생성합니다. 의도한 쉘 확장 바로 가기를 잘 모르겠으면 솔루션을 사용하여 학습과 연습을 진행합니다. 쉘 탭 완성을 사용하여 파일 경로 이름을 찾습니다. **/home/student** 디렉터리에서 이름이 **tv_seasonX_episodeY.ogg** 인 파일 12개를 생성합니다. 여기서 X는 시즌 번호, Y는 그 시즌의 에피소드로 바꿉니다. 각각 6개의 에피소드가 있는 두 시즌에 해당합니다.
4. 당신은 성공한 추리소설 시리즈 작가이며 출판할 다음 베스트셀러의 장을 편집하고 있습니다. 이름이 **mystery_chapterX.odf** 인 파일 8개를 생성합니다. 여기서 X는 1 ~ 8의 번호로 바꿉니다.
5. TV 에피소드를 구성하기 위해 단일 명령을 사용하여 **Videos** 디렉터리 아래에 **season1** 및 **season2**라는 하위 디렉터리 2개를 생성합니다. 적절한 TV 에피소드를 시즌 하위 디렉터리로 옮깁니다. 두 개의 명령만 사용하고 상대 경로 구문으로 대상을 지정합니다.
6. 단일 명령으로 2단계 디렉터리 계층 구조를 생성하여 추리소설 장을 구성합니다. **Documents** 디렉터리 아래에 **my_bestseller** 하위 디렉터리를 생성하고 새 **my_bestseller** 디렉터리 아래에 **chapters** 하위 디렉터리를 생성합니다. 단일 명령을 사용하여 **my_bestseller** 디렉터리 바로 아래에 하위 디렉터리를 3개 더 생성합니다. 이 하위 디렉터리의 이름을 각각 **editor**, **changes**, **vacation**이라고 짓습니다. **my_bestseller** 상위 디렉터리가 있으므로 **mkdir -p** 명령을 사용하여 상위를 생성할 필요는 없습니다.

7. **chapters** 디렉터리로 변경합니다. 틸드(~) 홈 디렉터리 바로 가기를 사용하여 책의 모든 장을 현재 디렉터리인 **chapters** 디렉터리로 이동합니다. 가장 간단한 구문을 사용하여 대상 디렉터리를 지정 합니다.
 검토를 위해 처음 두장을 편집자에게 전송하려고 합니다. 검토 중에 수정되지 않도록 이 두장만 **editor** 디렉터리로 이동합니다. **chapters** 하위 디렉터리부터 시작해서, 중괄호 확장에 범위를 사용하여 이동할 장 파일 이름과 대상 디렉터리의 상대 경로를 지정합니다.
 휴가 중에 7장과 8장을 쓰려고 합니다. 단일 명령을 사용하여 해당 파일을 **chapters** 디렉터리에서 **vacation** 디렉터리로 이동합니다. 와일드카드 문자를 사용하지 않고 중괄호 확장에 문자열 목록을 사용하여 장 파일 이름을 지정합니다.
8. 작업 디렉터리를 **~/Videos/season2**로 변경한 다음, 시즌의 첫 번째 에피소드를 **vacation** 디렉터리에 복사합니다. 단일 **cd** 명령을 사용하여 작업 디렉터리에서 **~/Documents/my_bestseller/vacation** 디렉터리로 변경합니다. 해당 파일을 나열합니다. 이전 작업 디렉터리 인수를 사용하여 **season2** 디렉터리로 돌아갑니다. (이 인수는 **cd** 명령을 사용한 마지막 디렉터리 변경이 여러 개의 **cd** 명령이 아니라 단일 명령만 사용했기 때문에 성공합니다.) **season2** 디렉터리의 에피소드 2 파일을 **vacation** 디렉터리에 복사합니다. 다시 바로 가기를 사용하여 **vacation** 디렉터리로 돌아갑니다.
9. 5장과 6장의 저자가 여러 가지 가능한 방식으로 변경해 보려고 합니다. 이러한 변경 중에 원본 파일이 수정되지 않도록 **~/Documents/my_bestseller/chapters** 디렉터리의 두 파일을 모두 **~/Documents/my_bestseller/changes** 디렉터리에 복사합니다. **~/Documents/my_bestseller** 디렉터리로 이동합니다. 대괄호 패턴 일치를 사용하여 **cp** 명령의 **filename** 인수에 일치시킬 장 번호를 지정합니다.
10. 현재 디렉터리를 **changes** 디렉터리로 변경하고 **date +%F** 명령에 명령 대체를 사용하여 **mystery_chapter5.odf** 파일을 전체 날짜가 포함된 새 파일에 복사합니다. **mystery_chapter5_YYYY-MM-DD.odf** 이름 형식을 사용합니다.
date +%s 명령에 명령 대체를 사용하여 **mystery_chapter5.odf**의 다른 복사본을 만들고 현재 타임스탬프(epoch인 1970-01-01 00:00 UTC 이후 경과한 초 수)를 추가하여 고유한 파일 이름을 지정합니다.
11. 추가 검토 후에 구성 변경이 불필요하다고 결정합니다. **changes** 디렉터리를 삭제합니다.
 필요한 경우 **changes** 디렉터리로 이동하여 디렉터리에 있는 모든 파일을 삭제합니다. 현재 작업 디렉터리인 경우에는 디렉터리를 삭제할 수 없습니다.
changes 디렉터리의 상위 디렉터리로 변경합니다. **-r** 재귀 옵션 없이 **rm** 명령을 사용하여 빈 디렉터리를 삭제해 봅니다. 이 시도는 실패할 것입니다. 마지막으로, **rmdir** 명령을 사용하여 빈 디렉터리를 삭제하면 성공합니다.
 휴가가 끝나면 **vacation** 디렉터리는 더 이상 필요하지 않습니다. **rm** 명령에 재귀 옵션을 사용하여 디렉터리를 삭제합니다.
 마쳤으면 **student** 사용자의 홈 디렉터리로 돌아갑니다.
12. **~/Documents/project_plans/season2_project_plan.odf** 파일에 대한 **~/Documents/backups/season2_project_plan.odf.back**이라는 하드 링크를 생성합니다. 하드 링크는 실수로 원본 파일이 삭제되는 것을 방지하며, 원본 파일이 변경될 경우 백업 파일을 계속 업데이트합니다.
 힌트: **~/Documents/backups** 디렉터리가 없는 경우 **mkdir** 명령을 사용하여 생성합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade files-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish files-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

명령줄에서 파일 관리

이 랩에서는 쉘과 다양한 파일 이름 일치 기술을 사용하여 파일과 디렉터리를 효율적으로 생성, 이동 및 제거합니다.

결과

- 와일드카드를 사용하여 파일을 찾아서 조작합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start files-review
```

지침

- ssh 명령을 사용하여 **student** 사용자로 **serverb** 시스템에 로그인합니다. 시스템 구성에 따라 인증을 위해 SSH 키를 사용할 수 있습니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
```

- Documents 디렉터리에 **project_plans**라는 디렉터리를 생성합니다. Documents 디렉터리는 **student** 사용자의 홈 디렉터리에 있어야 합니다. **project_plans** 디렉터리에 **season1_project_plan.odf** 및 **season2_project_plan.odf**라는 빈 파일 2개를 생성합니다.

힌트: ~/Documents 디렉터리가 없는 경우 **mkdir** 명령의 **-p** 옵션을 사용하여 생성합니다.

```
[student@serverb ~]$ mkdir -p ~/Documents/project_plans
[student@serverb ~]$ touch \
~/Documents/project_plans/{season1,season2}_project_plan.odf
[student@serverb ~]$ ls -lR Documents/
Documents/:
total 0
drwxr-xr-x. 2 student student 70 Mar  7 03:50 project_plans

Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:50 season1_project_plan.odf
-rw-r--r--. 1 student student 0 Mar  7 03:50 season2_project_plan.odf
```

- 이 랩에서 사용할 빈 연습 파일 세트를 생성합니다. 의도한 쉘 확장 바로 가기를 잘 모르겠으면 솔루션을 사용하여 학습과 연습을 진행합니다. 쉘 탭 완성을 사용하여 파일 경로 이름을 찾습니다. /home/

2장 | 명령줄에서 파일 관리

student 디렉터리에서 이름이 **tv_seasonX_episodeY.ogg** 인 파일 12개를 생성합니다. 여기서 X는 시즌 번호, Y는 그 시즌의 에피소드로 바꿉니다. 각각 6개의 에피소드가 있는 두 시즌에 해당합니다.

```
[student@serverb ~]$ touch tv_season{1..2}_episode{1..6}.ogg
[student@serverb ~]$ ls tv*
tv_season1_episode1.ogg  tv_season1_episode5.ogg  tv_season2_episode3.ogg
tv_season1_episode2.ogg  tv_season1_episode6.ogg  tv_season2_episode4.ogg
tv_season1_episode3.ogg  tv_season2_episode1.ogg  tv_season2_episode5.ogg
tv_season1_episode4.ogg  tv_season2_episode2.ogg  tv_season2_episode6.ogg
```

- 당신은 성공한 추리소설 시리즈 작가이며 출판할 다음 베스트셀러의 장을 편집하고 있습니다. 이름이 **mystery_chapterX.odf** 인 파일 8개를 생성합니다. 여기서 X는 1~8의 번호로 바꿉니다.

```
[student@serverb ~]$ touch mystery_chapter{1..8}.odf
[student@serverb ~]$ ls mys*
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.odf
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.odf
mystery_chapter3.odf  mystery_chapter6.odf
```

- TV 에피소드를 구성하기 위해 단일 명령을 사용하여 **Videos** 디렉터리 아래에 **season1** 및 **season2**라는 하위 디렉터리 2개를 생성합니다. 적절한 TV 에피소드를 시즌 하위 디렉터리로 옮깁니다. 두 개의 명령만 사용하고 상대 경로 구문으로 대상을 지정합니다.

- 단일 명령을 사용하여 **Videos** 디렉터리 아래에 **season1** 및 **season2**라는 하위 디렉터리 2개를 생성합니다.

```
[student@serverb ~]$ mkdir -p Videos/season{1..2}
[student@serverb ~]$ ls Videos
season1  season2
```

- 두 개의 명령만 사용하여 적절한 TV 에피소드를 시즌 하위 디렉터리로 이동합니다.

```
[student@serverb ~]$ mv tv_season1* Videos/season1
[student@serverb ~]$ mv tv_season2* Videos/season2
[student@serverb ~]$ ls -R Videos
Videos:
season1  season2

Videos/season1:
tv_season1_episode1.ogg  tv_season1_episode3.ogg  tv_season1_episode5.ogg
tv_season1_episode2.ogg  tv_season1_episode4.ogg  tv_season1_episode6.ogg

Videos/season2:
tv_season2_episode1.ogg  tv_season2_episode3.ogg  tv_season2_episode5.ogg
tv_season2_episode2.ogg  tv_season2_episode4.ogg  tv_season2_episode6.ogg
```

- 단일 명령으로 2단계 디렉터리 계층 구조를 생성하여 추리소설 장을 구성합니다. **Documents** 디렉터리 아래에 **my_bestseller** 하위 디렉터리를 생성하고 새 **my_bestseller** 디렉터리 아래에 **chapters** 하위 디렉터리를 생성합니다. 단일 명령을 사용하여 **my_bestseller** 디렉터리 바로 아래에 하위 디렉터리를 3개 더 생성합니다. 이 하위 디렉터리의 이름을 각각 **editor**, **changes**, **vacation**이라고 짓습니다. **my_bestseller** 상위 디렉터리가 있으므로 **mkdir -p** 명령을 사용하여 상위를 생성할 필요는 없습니다.

- 6.1. **Documents** 디렉터리 아래에 **my_bestseller** 디렉터리를 생성합니다. **my_bestseller** 디렉터리 아래에 **chapters** 디렉터리를 생성합니다.

```
[student@serverb ~]$ mkdir -p Documents/my_bestseller/chapters
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
chapters

Documents/my_bestseller/chapters:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
```

- 6.2. 단일 명령을 사용하여 **my_bestseller** 디렉터리 아래에 **editor**, **changes**, **vacation**이라는 디렉터리 3개를 생성합니다.

```
[student@serverb ~]$ mkdir Documents/my_bestseller/{editor,changes,vacation}
[student@serverb ~]$ ls -R Documents
Documents:
my_bestseller  project_plans

Documents/my_bestseller:
changes  chapters  editor  vacation

Documents/my_bestseller/changes:

Documents/my_bestseller/chapters:

Documents/my_bestseller/editor:

Documents/my_bestseller/vacation:

Documents/project_plans:
season1_project_plan.odf  season2_project_plan.odf
```

7. **chapters** 디렉터리로 변경합니다. 틸드(~) 홈 디렉터리 바로 가기를 사용하여 책의 모든 장을 현재 디렉터리인 **chapters** 디렉터리로 이동합니다. 가장 간단한 구문을 사용하여 대상 디렉터리를 지정합니다.

검토를 위해 처음 두장을 편집자에게 전송하려고 합니다. 검토 중에 수정되지 않도록 이 두장만 **editor** 디렉터리로 이동합니다. **chapters** 하위 디렉터리부터 시작해서, 중괄호 확장에 범위를 사용하여 이동할 장 파일 이름과 대상 디렉터리의 상대 경로를 지정합니다.

휴가 중에 7장과 8장을 쓰려고 합니다. 단일 명령을 사용하여 해당 파일을 **chapters** 디렉터리에서 **vacation** 디렉터리로 이동합니다. 와일드카드 문자를 사용하지 않고 중괄호 확장에 문자열 목록을 사용하여 장 파일 이름을 지정합니다.

- 7.1. **chapters** 디렉터리로 변경하고 틸드(~) 홈 디렉터리 바로 가기를 사용하여 책의 모든 장을 **chapters** 디렉터리로 이동합니다.

```
[student@serverb ~]$ cd Documents/my_bestseller/chapters
[student@serverb chapters]$ mv ~/mystery_chapter* .
[student@serverb chapters]$ ls
mystery_chapter1.odf  mystery_chapter4.odf  mystery_chapter7.odf
mystery_chapter2.odf  mystery_chapter5.odf  mystery_chapter8.odf
mystery_chapter3.odf  mystery_chapter6.odf
```

- 7.2. 처음 두 장을 **editor** 디렉터리로 이동합니다. 중괄호 확장에 범위를 사용하여 이동할 장 파일 이름과 대상 디렉터리의 상대 경로를 지정합니다.

```
[student@serverb chapters]$ mv mystery_chapter{1..2}.odf ..../editor
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf  mystery_chapter7.odf
mystery_chapter4.odf  mystery_chapter6.odf  mystery_chapter8.odf
[student@serverb chapters]$ ls ..../editor
mystery_chapter1.odf  mystery_chapter2.odf
```

- 7.3. 단일 명령을 사용하여 7장과 8장을 **chapters** 디렉터리에서 **vacation** 디렉터리로 이동합니다. 와일드카드 문자를 사용하지 않고 중괄호 확장에 문자열 목록을 사용하여 장 파일 이름을 지정합니다.

```
[student@serverb chapters]$ mv mystery_chapter{7,8}.odf ..../vacation
[student@serverb chapters]$ ls
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb chapters]$ ls ..../vacation
mystery_chapter7.odf  mystery_chapter8.odf
```

8. 작업 디렉터리를 **~/Videos/season2**로 변경한 다음, 시즌의 첫 번째 에피소드를 **vacation** 디렉터리에 복사합니다. 단일 **cd** 명령을 사용하여 작업 디렉터리에서 **~/Documents/my_bestseller/vacation** 디렉터리로 변경합니다. 해당 파일을 나열합니다. 이전 작업 디렉터리 인수를 사용하여 **season2** 디렉터리로 돌아갑니다. (이 인수는 **cd** 명령을 사용한 마지막 디렉터리 변경이 여러 개의 **cd** 명령이 아니라 단일 명령만 사용했기 때문에 성공합니다.) **season2** 디렉터리의 에피소드 2 파일을 **vacation** 디렉터리에 복사합니다. 다시 바로 가기를 사용하여 **vacation** 디렉터리로 돌아갑니다.

- 8.1. 작업 디렉터리를 **~/Videos/season2**로 변경한 다음, 시즌의 첫 번째 에피소드를 **vacation** 디렉터리에 복사합니다.

```
[student@serverb chapters]$ cd ~/Videos/season2
[student@serverb season2]$ cp *episode1.ogg ~/Documents/my_bestseller/vacation
```

- 8.2. 단일 **cd** 명령을 사용하여 작업 디렉터리에서 **~/Documents/my_bestseller/vacation** 디렉터리로 변경하고, 해당 파일을 표시하고, **-** 인수를 사용하여 이전 디렉터리로 돌아갑니다. 에피소드 2 파일을 **vacation** 디렉터리에 복사합니다. **cd** 명령에 **-** 인수를 사용하여 **vacation** 디렉터리로 돌아갑니다.

```
[student@serverb season2]$ cd ~/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf  mystery_chapter8.odf  tv_season2_episode1.ogg
```

2장 | 명령줄에서 파일 관리

```
[student@serverb vacation]$ cd -
/home/student/Videos/season2
[student@serverb season2]$ cp *episode2.ogg ~/Documents/my_bestseller/vacation
[student@serverb season2]$ cd -
/home/student/Documents/my_bestseller/vacation
[student@serverb vacation]$ ls
mystery_chapter7.odf  tv_season2_episode1.ogg
mystery_chapter8.odf  tv_season2_episode2.ogg
```

9. 5장과 6장의 저자가 여러 가지 가능한 방식으로 변경해 보려고 합니다. 이러한 변경 중에 원본 파일이 수정되지 않도록 **~/Documents/my_bestseller/chapters** 디렉터리의 두 파일을 모두 **~/Documents/my_bestseller/changes** 디렉터리에 복사합니다. **~/Documents/my_bestseller** 디렉터리로 이동합니다. 대괄호 패턴 일치를 사용하여 **cp** 명령의 **filename** 인수에 일치시킬 장 번호를 지정합니다.

```
[student@serverb vacation]$ cd ~/Documents/my_bestseller
[student@serverb my_bestseller]$ cp chapters/mystery_chapter[56].odf changes
[student@serverb my_bestseller]$ ls chapters
mystery_chapter3.odf  mystery_chapter5.odf
mystery_chapter4.odf  mystery_chapter6.odf
[student@serverb my_bestseller]$ ls changes
mystery_chapter5.odf  mystery_chapter6.odf
```

10. 현재 디렉터리를 **changes** 디렉터리로 변경하고 **date +%F** 명령에 명령 대체를 사용하여 **mystery_chapter5.odf** 파일을 전체 날짜가 포함된 새 파일에 복사합니다. **mystery_chapter5_YYYY-MM-DD.odf** 이름 형식을 사용합니다. **date +%s** 명령에 명령 대체를 사용하여 **mystery_chapter5.odf**의 다른 복사본을 만들고 현재 타임스탬프(epoch인 1970-01-01 00:00 UTC 이후 경과한 초 수)를 추가하여 고유한 파일 이름을 지정합니다.

```
[student@serverb my_bestseller]$ cd changes
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%F).odf
[student@serverb changes]$ cp mystery_chapter5.odf \
mystery_chapter5_$(date +%s).odf
[student@serverb changes]$ ls
mystery_chapter5_1646644424.odf  mystery_chapter5.odf
mystery_chapter5_2022-03-07.odf  mystery_chapter6.odf
```

11. 추가 검토 후에 구성 변경이 불필요하다고 결정합니다. **changes** 디렉터리를 삭제합니다. 필요한 경우 **changes** 디렉터리로 이동하여 디렉터리에 있는 모든 파일을 삭제합니다. 현재 작업 디렉터리인 경우에는 디렉터리를 삭제할 수 없습니다. **changes** 디렉터리의 상위 디렉터리로 변경합니다. **-r** 재귀 옵션 없이 **rm** 명령을 사용하여 빈 디렉터리를 삭제해 봅니다. 이 시도는 실패할 것입니다. 마지막으로, **rmdir** 명령을 사용하여 빈 디렉터리를 삭제하면 성공합니다. 휴가가 끝나면 **vacation** 디렉터리는 더 이상 필요하지 않습니다. **rm** 명령에 재귀 옵션을 사용하여 디렉터리를 삭제합니다. 마쳤으면 **student** 사용자의 홈 디렉터리로 돌아갑니다.
- 11.1. **changes** 디렉터리를 삭제합니다. **changes** 디렉터리의 상위 디렉터리로 변경하고 **-r** 재귀 옵션 없이 **rm** 명령을 사용하여 빈 디렉터리를 삭제하면 실패합니다. **rmdir** 명령을 사용하여 빈 디렉터리를 삭제합니다.

```
[student@serverb changes]$ rm mystery*
[student@serverb changes]$ cd ..
[student@serverb my_bestseller]$ rm changes
rm: cannot remove 'changes': Is a directory
[student@serverb my_bestseller]$ rmdir changes
[student@serverb my_bestseller]$ ls
chapters editor vacation
```

11.2. **rm** 명령에 **-r** 옵션을 사용하여 **vacation** 디렉터리를 삭제합니다. **student** 사용자의 홈 디렉터리로 돌아갑니다.

```
[student@serverb my_bestseller]$ rm -r vacation
[student@serverb my_bestseller]$ ls
chapters editor
[student@serverb my_bestseller]$ cd
[student@serverb ~]$
```

12. **~/Documents/project_plans/season2_project_plan.odf** 파일에 대한 **~/Documents/backups/season2_project_plan.odf.back**이라는 하드 링크를 생성합니다. 하드 링크는 실수로 원본 파일이 삭제되는 것을 방지하며, 원본 파일이 변경될 경우 백업 파일을 계속 업데이트합니다.

힌트: **~/Documents/backups** 디렉터리가 없는 경우 **mkdir** 명령을 사용하여 생성합니다.

- 12.1. **~/Documents/project_plans/season2_project_plan.odf** 파일에 대한 **~/Documents/backups/season2_project_plan.odf.back**이라는 하드 링크를 생성합니다.

```
[student@serverb ~]$ mkdir ~/Documents/backups
[student@serverb ~]$ ln ~/Documents/project_plans/season2_project_plan.odf \
~/Documents/backups/season2_project_plan.odf.back
[student@serverb ~]$ ls -lR ~/Documents/
/home/student/Documents/:
total 0
drwxr-xr-x. 2 student student 43 Mar  7 04:18 backups
drwxr-xr-x. 4 student student 36 Mar  7 04:16 my_bestseller
drwxr-xr-x. 2 student student 70 Mar  7 03:50 project_plans

/home/student/Documents/backups:
total 0
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_plan.odf.back

/home/student/Documents/my_bestseller:
total 0
drwxr-xr-x. 2 student student 118 Mar  7 04:07 chapters
drwxr-xr-x. 2 student student 62 Mar  7 04:06 editor

/home/student/Documents/my_bestseller/chapters:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter3.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter4.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter5.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter6.odf
```

```
/home/student/Documents/my_bestseller/editor:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter1.odf
-rw-r--r--. 1 student student 0 Mar  7 03:56 mystery_chapter2.odf

/home/student/Documents/project_plans:
total 0
-rw-r--r--. 1 student student 0 Mar  7 03:50 season1_project_plan.odf
-rw-r--r--. 2 student student 0 Mar  7 03:50 season2_project_plan.odf
```

2 및 `season2_project_plan.odf.back` 파일 둘 다에서 링크 수는 `season2_project_plan.odf`입니다.

12.2. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade files-review
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish files-review
```

이것으로 섹션을 완료합니다.

요약

- Linux 시스템의 파일은 파일 시스템 계층 구조라는 반전된 단일 디렉터리 트리로 구성됩니다.
- 절대 경로는 슬래시 문자(/)로 시작하며 파일 시스템 계층 구조의 파일 위치를 지정합니다.
- 상대 경로는 슬래시 문자로 시작하지 않습니다.
- 상대 경로는 현재 작업 디렉터리에 상대적인 파일 위치를 지정합니다.
- 점(..), 이중 점(..), 틸드(~) 특수 문자와 함께 명령을 사용하여 파일 시스템의 파일 위치를 나타낼 수 있습니다.
- **mkdir, rmdir, cp, mv, rm** 명령은 Linux에서 파일을 관리하는 주요 명령입니다.
- 하드 링크와 소프트 링크는 여러 개의 파일 이름이 동일한 데이터를 가리키는 서로 다른 방법입니다.
- Bash 쉘은 효율적으로 명령을 실행할 수 있도록 패턴 일치, 확장 및 대체 기능을 제공합니다.

3장

로컬 사용자 및 그룹 관리

목적

로컬 사용자 및 그룹을 생성, 관리, 삭제하고 로컬 암호 정책을 관리합니다.

목표

- Linux 시스템에 있는 사용자 및 그룹의 목적을 설명합니다.
- 수퍼유저 계정으로 전환하여 Linux 시스템을 관리하고, `sudo` 명령을 통해 다른 사용자에게 수퍼유저 액세스 권한을 부여합니다.
- 로컬 사용자 계정을 생성, 관리 및 삭제합니다.
- 로컬 그룹 계정을 생성, 수정 및 삭제합니다.
- 사용자에 대한 암호 관리 정책을 설정하고, 수동으로 사용자 계정을 잠그거나 잠금을 해제합니다.

섹션

- 사용자 및 그룹 개념(및 퀴즈)
- 수퍼유저 액세스 권한 얻기(안내에 따른 연습)
- 로컬 사용자 계정 관리(안내에 따른 연습)
- 로컬 그룹 계정 관리(안내에 따른 연습)
- 사용자 암호 관리(안내에 따른 연습)

랩

로컬 사용자 및 그룹 관리

사용자 및 그룹 개념 설명

목표

Linux 시스템에 있는 사용자 및 그룹의 목적을 설명합니다.

사용자란?

사용자 계정은 사용자와 명령을 실행할 수 있는 프로그램 간에 보안 경계를 제공합니다.

일반 사용자로 확인되고 작업하기 쉽도록 사용자에는 사용자 이름이 있습니다. 내부적으로 시스템은 할당된 고유 식별 번호, 사용자 ID 또는 UID로 사용자 계정을 구별합니다. 대부분의 시나리오에서 사용자가 사용자 계정을 사용하는 경우 시스템은 로그인할 수 있는 권한이 있는 사용자임을 입증하기 위해 사용자에게 비밀 암호를 할당합니다.

사용자 계정은 시스템 보안의 기본 요소입니다. 시스템의 모든 프로세스(실행 프로그램)는 특정 사용자로 실행됩니다. 모든 파일에는 특정 사용자가 소유자로 지정되어 있습니다. 시스템은 파일 소유권을 사용하여 파일 사용자에 대해 액세스 제어를 적용합니다. 실행 중인 프로세스와 관련된 사용자에 따라 해당 프로세스에서 액세스할 수 있는 파일과 디렉터리가 결정됩니다.

사용자 계정의 기본 유형은 수퍼유저, 시스템 사용자, 일반 사용자입니다.

- 수퍼유저 계정은 시스템을 관리합니다. 수퍼유저 이름은 **root**이고 계정의 UID는 0입니다. 수퍼유저는 모든 시스템 액세스 권한을 갖습니다.
- 시스템 사용자 계정은 지원 서비스를 제공하는 프로세스에서 사용됩니다. 이러한 프로세스 또는 데몬은 일반적으로 수퍼유저로 실행할 필요가 없습니다. 다른 시스템 사용자와 시스템의 일반 사용자로부터 파일 및 기타 리소스를 보호하기 위해 시스템 사용자에게는 권한 없는 계정이 할당됩니다. 사용자는 시스템 사용자 계정을 사용하여 대화형으로 로그인하지 않습니다.
- 대부분의 사용자는 일상적인 작업에 사용되는 일반 사용자 계정을 보유합니다. 시스템 사용자와 마찬가지로 일반 사용자는 시스템에 대한 액세스가 제한됩니다.

id 명령을 사용하여 현재 로그인한 사용자에 대한 정보를 표시합니다.

```
[user01@host ~]$ id
uid=1000(user01) gid=1000(user01) groups=1000(user01)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

다른 사용자에 대한 정보를 보려면 사용자 이름을 **id** 명령에 인수로 전달합니다.

```
[user01@host ~]$ id user02
uid=1002(user02) gid=1001(user02) groups=1001(user02)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

파일의 소유자를 보려면 **ls -l** 명령을 사용합니다. 디렉터리 내용이 아니라 디렉터리 소유자를 보려면 **ls -ld** 명령을 사용합니다. 다음 출력에서는 세 번째 열에 사용자 이름이 표시됩니다.

```
[user01@host ~]$ ls -l mytextfile.txt
-rw-rw-r-- 1 user01 user01 0 Feb 5 11:10 mytextfile.txt
[user01@host]$ ls -ld Documents
drwxrwxr-x. 2 user01 user01 6 Feb 5 11:10 Documents
```

프로세스 정보를 보려면 **ps** 명령을 사용합니다. 기본값은 현재 쉘의 프로세스만 표시하는 것입니다. 터미널에서 모든 프로세스를 보려면 **ps** 명령의 **-a** 옵션을 사용합니다. 프로세스와 관련된 사용자를 보려면 **ps** 명령의 **-u** 옵션을 사용합니다. 다음 출력에서는 첫 번째 열에 사용자 이름이 표시됩니다.

```
[user01@host ~]$ ps -au
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START  TIME COMMAND
root     1690  0.0  0.0 220984 1052  ttyp0  Ss+  22:43  0:00 /sbin/agetty -o -p --
\u --keep-baud 1
user01  1769  0.0  0.1 377700 6844  ttyp2  Ssl+ 22:45  0:00 /usr/libexec/gdm-x-
session --register-
user01  1773  1.3  1.3 528948 78356  ttyp2  Sl+   22:45  0:03 /usr/libexec/Xorg vt2
-displayfd 3 -au
user01  1800  0.0  0.3 521412 19824  ttyp2  Sl+   22:45  0:00 /usr/libexec/gnome-
session-binary
user01  3072  0.0  0.0 224152 5756  pts/1   Ss    22:48  0:00 -bash
user01  3122  0.0  0.0 225556 3652  pts/1   R+   22:49  0:00 ps -au
```

이전 명령의 출력에서는 사용자가 이름으로 표시되지만, 내부적으로 운영 체제는 UID를 사용하여 사용자를 추적합니다. 사용자 이름과 UID의 매핑은 계정 정보의 데이터베이스에서 정의됩니다. 기본적으로 시스템은 **/etc/passwd** 파일을 사용하여 로컬 사용자에 대한 정보를 저장합니다.

/etc/passwd 파일에 있는 각 행에는 한 명의 사용자에 대한 정보가 포함되어 있습니다. 파일은 콜론으로 구분된 7개의 필드로 나뉘어져 있습니다. **/etc/passwd**의 줄 예제는 다음과 같습니다.

```
[user01@host ~]$ cat /etc/passwd
...output omitted...
user01:x:1000:1000:User One:/home/user01:/bin/bash
```

콜론으로 구분된 코드 블록의 각 부분을 살펴보십시오.

- **user01**: 이 사용자의 사용자 이름입니다.
- **x**: 사용자의 암호화된 암호가 여기에 저장되었습니다. 이것은 이제 자리 표시자입니다.
- **1000**: 이 사용자 계정의 UID 번호입니다.
- **1000**: 이 사용자 계정의 기본 그룹에 대한 GID 번호입니다. 그룹에 대해서는 이 섹션의 뒷부분에서 설명합니다.
- **User One**: 이 사용자에 대한 간단한 주석, 설명 또는 실제 이름입니다.
- **/home/user01**: 사용자의 홈 디렉터리 및 로그인 쉘이 시작될 때의 초기 작업 디렉터리입니다.
- **/bin/bash**: 로그인 시 실행되는 이 사용자의 기본 쉘 프로그램입니다. 일부 계정은 **/sbin/nologin** 쉘을 통해 해당 계정을 사용한 대화형 로그인을 허용하지 않습니다.

그룹이란?

그룹은 파일 및 기타 시스템 리소스에 대한 액세스를 공유해야 하는 사용자 컬렉션입니다. 그룹은 단일 사용자 대신 사용자 세트에 파일 액세스 권한을 부여할 수 있습니다.

사용자와 마찬가지로 그룹에는 인식하기 쉽도록 그룹 이름이 있습니다. 내부적으로 시스템은 할당된 고유 식별 번호, 그룹 ID 또는 GID로 그룹을 구별합니다. 그룹 이름과 GID 매핑은 그룹 계정 정보의 ID 관리 데이터베

3장 | 로컬 사용자 및 그룹 관리

이스에서 정의됩니다. 기본적으로 시스템은 **/etc/group** 파일을 사용하여 로컬 그룹에 대한 정보를 저장합니다.

/etc/group 파일에 있는 각 행에는 하나의 그룹에 대한 정보가 포함되어 있습니다. 각 그룹 항목은 콜론으로 구분된 4개의 필드로 나뉩니다. **/etc/group**의 줄 예제는 다음과 같습니다.

```
[user01@host ~]$ cat /etc/group
...output omitted...
group01:x:10000:user01,user02,user03
```

콜론으로 구분된 코드 블록의 각 부분을 살펴보십시오.

- **group01** : 이 그룹의 이름입니다.
- **x** : 사용되지 않는 그룹 암호 필드이며 지금은 자리 표시자입니다.
- **10000** : 이 그룹의 GID 번호(**10000**)입니다.
- **user01,user02,user03** : 보조 그룹으로 이 그룹의 멤버인 사용자 목록입니다.

기본 그룹 및 보조 그룹

모든 사용자는 정확히 하나의 기본 그룹을 갖고 있습니다. 로컬 사용자의 경우 이 그룹은 **/etc/passwd** 파일에 GID로 표시됩니다. 기본 그룹은 사용자가 생성한 파일을 소유합니다.

일반 사용자를 생성하는 경우 사용자의 기본 그룹이 될 그룹이 사용자와 동일한 이름으로 생성됩니다. 사용자는 이 사용자 개인 그룹의 유일한 멤버입니다. 이 그룹 멤버십 설계는 사용자 그룹이 기본적으로 분리되도록 파일 권한 관리를 간소화합니다.

사용자에게는 보조 그룹이 있을 수도 있습니다. 보조 그룹의 멤버십은 **/etc/group** 파일에 저장됩니다. 기본 그룹이든, 보조 그룹이든 간에 해당 그룹에 액세스 권한이 있는지에 따라 사용자에게 파일 액세스 권한이 부여됩니다. 예를 들어 **user01** 사용자의 기본 그룹이 **user01**이고 보조 그룹은 **wheel** 및 **webadmin**인 경우, 사용자는 세 그룹 중 하나라도 읽기 가능한 파일이라면 해당 파일을 읽을 수 있습니다.

id 명령은 사용자의 그룹 멤버십을 표시할 수 있습니다. 다음 예제에서 **user01** 사용자의 기본 그룹(**gid**)은 **user01** 그룹입니다. **groups** 항목은 이 사용자의 모든 그룹 멤버십을 표시하며, 사용자의 보조 그룹으로 **wheel** 및 **group01** 그룹도 있습니다.

```
[user01@host ~]$ id
uid=1001(user01) gid=1003(user01) groups=1003(user01),10(wheel),10000(group01)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```



참조

id(1), **passwd(5)** 및 **group(5)** 도움말 페이지

info libc(GNU C Library Reference Manual)

- 섹션 30: 사용자 및 그룹

(이 **info** 노드를 사용할 수 있으려면 **glibc-devel** 패키지를 설치해야 합니다.)

▶ 퀴즈

사용자 및 그룹 개념 설명

다음 질문에 대해 올바른 답을 선택하십시오.

▶ 1. 다음 중 가장 기본적인 수준에서 사용자를 식별하는 번호를 나타내는 항목은 무엇입니까?

- a. 기본 사용자
- b. UID
- c. GID
- d. 사용자 이름

▶ 2. 다음 중 사용자의 명령줄 프롬프트를 제공하는 프로그램을 나타내는 항목은 무엇입니까?

- a. 기본 쉘
- b. 홈 디렉터리
- c. 로그인 쉘
- d. 명령 이름

▶ 3. 다음 중 로컬 그룹 정보의 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. /etc/passwd
- c. /etc/GID
- d. /etc/group

▶ 4. 다음 중 사용자의 개인 파일 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. 로그인 쉘
- c. /etc/passwd
- d. /etc/group

▶ 5. 다음 중 가장 기본적인 수준에서 그룹을 식별하는 번호를 나타내는 항목은 무엇입니까?

- a. 기본 그룹
- b. UID
- c. GID
- d. Groupid

▶ 6. 다음 중 로컬 사용자 계정 정보의 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. /etc/passwd
- c. /etc/UID
- d. /etc/group

▶ 7. /etc/passwd 파일의 네 번째 필드는 무엇입니까?

- a. 홈 디렉터리
- b. UID
- c. 로그인 쉘
- d. 기본 그룹

▶ 솔루션

사용자 및 그룹 개념 설명

다음 질문에 대해 올바른 답을 선택하십시오.

▶ 1. 다음 중 가장 기본적인 수준에서 사용자를 식별하는 번호를 나타내는 항목은 무엇입니까?

- a. 기본 사용자
- b. UID
- c. GID
- d. 사용자 이름

▶ 2. 다음 중 사용자의 명령줄 프롬프트를 제공하는 프로그램을 나타내는 항목은 무엇입니까?

- a. 기본 쉘
- b. 홈 디렉터리
- c. 로그인 쉘
- d. 명령 이름

▶ 3. 다음 중 로컬 그룹 정보의 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. /etc/passwd
- c. /etc/GID
- d. /etc/group

▶ 4. 다음 중 사용자의 개인 파일 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. 로그인 쉘
- c. /etc/passwd
- d. /etc/group

▶ 5. 다음 중 가장 기본적인 수준에서 그룹을 식별하는 번호를 나타내는 항목은 무엇입니까?

- a. 기본 그룹
- b. UID
- c. GID
- d. Groupid

▶ 6. 다음 중 로컬 사용자 계정 정보의 위치를 나타내는 항목 또는 파일은 무엇입니까?

- a. 홈 디렉터리
- b. /etc/passwd
- c. /etc/UID
- d. /etc/group

▶ 7. /etc/passwd 파일의 네 번째 필드는 무엇입니까?

- a. 홈 디렉터리
- b. UID
- c. 로그인 쉘
- d. 기본 그룹

수퍼유저 액세스 권한 얻기

목표

수퍼유저 계정으로 전환하여 Linux 시스템을 관리하고, **sudo** 명령을 통해 다른 사용자에게 수퍼유저 액세스 권한을 부여합니다.

수퍼유저

대부분의 운영 체제에는 시스템에 대한 모든 권한을 가진 수퍼유저가 있습니다. Red Hat Enterprise Linux에서 수퍼유저는 **root** 사용자입니다. 이 사용자는 파일 시스템에 대한 일반 권한을 재정의할 수 있으며 시스템을 관리하는 데 사용될 수 있습니다. 소프트웨어 설치 또는 제거와 같은 작업을 수행하고 시스템 파일 및 디렉터리를 관리하려면 사용자 권한을 **root** 사용자로 에스컬레이션해야 합니다.

일반적으로 **root** 사용자만 대부분의 장치를 제어할 수 있지만 몇 가지 예외가 적용됩니다. 일반 사용자는 USB 장치와 같은 휴대용 장치를 제어할 수 있습니다. 따라서 일반 사용자가 파일을 추가 및 제거하고 다른 방식으로 휴대용 장치를 관리할 수 있지만 **root** 사용자만 기본적으로 하드 드라이브를 관리할 수 있습니다.

하지만 이러한 무제한 권한에는 책임이 따릅니다. **root** 사용자는 시스템을 손상시킬 수 있는 무제한 능력을 갖습니다. 파일과 디렉터리를 제거하고, 사용자 계정을 제거하고, 백도어를 추가할 수 있습니다. **root** 사용자 계정이 손상되면 시스템이 위험해지며 관리 권한을 잃을 수 있습니다. 시스템 관리자는 항상 일반 사용자로 로그인하고 필요한 경우에만 권한을 **root**로 에스컬레이션하는 것이 좋습니다.

Linux의 **root** 계정은 Microsoft Windows의 로컬 **Administrator** 계정과 비슷합니다. Linux에서는 대부분의 시스템 관리자가 권한이 없는 사용자로 시스템에 로그인한 후 다양한 도구를 사용하여 임시로 **root** 권한을 얻습니다.



경고

Microsoft Windows 사용자는 로컬 **Administrator** 사용자로 로그인하여 시스템 관리자 역할을 수행하는 방식에 익숙할 수도 있습니다. 현재 이 사례는 권장되지 않으며, 사용자는 **Administrators** 그룹의 멤버십을 통해 관리 작업을 수행할 수 있는 권한을 얻습니다. 마찬가지로 RHEL에서는 시스템 관리자가 **root**로 직접 로그인하지 않는 것이 좋습니다. 대신, 시스템 관리자는 일반 사용자로 로그인한 후 다른 메커니즘(예: **su**, **sudo** 또는 **PolicyKit**)을 사용하여 수퍼유저 권한을 임시로 얻습니다.

root로 로그인하면 전체 데스크탑 환경이 불필요하게 관리 권한으로 실행됩니다. 일반적으로 일반 사용자 계정만 손상시킬 수 있는 보안 취약점으로 인해 잠재적으로 전체 시스템이 손상될 수 있습니다.

사용자 계정 전환

su 명령을 사용하면 사용자가 다른 사용자 계정으로 전환할 수 있습니다. 다른 사용자 계정을 매개 변수로 사용하여 일반 사용자 계정에서 **su** 명령을 실행하는 경우 전환할 대상 계정의 암호를 제공해야 합니다. **root** 사용자가 **su** 명령을 실행하는 경우 사용자의 암호를 입력할 필요는 없습니다.

이 예제에서는 **user01** 계정에서 **su** 명령을 사용하여 **user02** 계정으로 전환합니다.

```
[user01@host ~]$ su - user02
Password: user02_password
[user02@host ~]$
```

사용자 이름을 생략하면 **su** 또는 **su -** 명령은 기본적으로 **root**로 전환하려고 합니다.

```
[user01@host ~]$ su -
Password: root_password
[root@host ~]#
```

su 명령은 비로그인 쉘을 시작하는 반면, **su -** 명령(대시 옵션 포함)은 로그인 쉘을 시작합니다. 두 명령의 주요 차이점은 **su -**는 해당 사용자로 새로 로그인한 것처럼 쉘 환경을 설정하는 반면, **su**는 해당 사용자로 쉘을 시작하지만 원래 사용자의 환경 설정을 사용한다는 것입니다.

대부분의 경우 관리자는 **su -**를 실행하여 타겟 사용자의 일반 환경 설정으로 쉘을 가져와야 합니다. 자세한 내용은 **bash(1)** 도움말 페이지를 참조하십시오.



참고

su 명령은 다른 사용자(일반적으로 **root** 사용자)로 실행되는 명령줄 인터페이스(쉘 프롬프트)를 가져오는데 주로 사용됩니다. 그러나 **su** 명령의 **-c** 옵션을 사용하면 다른 사용자로 임의 프로그램을 실행할 수 있습니다. 이 동작은 Windows **runas** 유ти리티와 비슷합니다. 자세한 내용을 보려면 **info su**를 실행합니다.

Sudo를 사용하여 명령 실행

보안상의 이유로 시스템 관리자가 유효한 암호를 갖지 않도록 **root** 사용자를 구성하는 경우가 있습니다. 따라서 사용자가 암호를 사용하여 **root**로 직접 시스템에 로그인할 수 없습니다. 또한 **su**를 사용하여 대화형 쉘을 가져올 수 없습니다. 이 경우 **sudo** 명령을 사용하여 **root** 액세스 권한을 얻을 수 있습니다.

su 명령과 달리, **sudo**를 사용하는 경우 일반적으로 사용자가 액세스하려는 사용자 계정의 암호가 아닌 본인 암호를 입력하여 인증을 받아야 합니다. 즉, **sudo** 명령을 사용하여 **root**로 명령을 실행하는 사용자는 **root** 암호를 몰라도 됩니다. 대신 고유 암호를 사용하여 액세스를 인증합니다.

다음 표에는 **su**, **su -**, **sudo** 명령의 차이점이 요약되어 있습니다.

	su	su -	sudo
새 사용자 되기	예	예	에스컬레이션된 명령 기준
환경	현재 사용자	새 사용자	현재 사용자
암호 필요	새 사용자	새 사용자	현재 사용자
권한	새 사용자와 동일	새 사용자와 동일	구성에서 정의
활동 기록	su 명령만	su 명령만	에스컬레이션된 명령 기준

또한 특정 사용자가 모든 명령을 다른 사용자로 실행하거나 일부 명령만 해당 사용자로 실행할 수 있도록 **sudo** 명령을 구성할 수 있습니다. 예를 들어 **user01** 사용자가 **root**로 **usermod** 명령을 실행할 수 있도록 **sudo** 명령을 구성한 경우 다음 명령을 실행하여 사용자 계정을 잠그거나 잠금 해제할 수 있습니다.

```
[user01@host ~]$ sudo usermod -L user02
[sudo] password for user01: user01_password
[user01@host ~]$ su - user02
Password: user02_password
su: Authentication failure
[user01@host ~]$
```

사용자가 명령을 다른 사용자로 실행하려고 하는데 **sudo** 구성에서 허용하지 않는 경우 Bash는 명령을 차단하고 시도를 기록한 다음, 기본적으로 **root** 사용자에게 이메일을 전송합니다.

```
[user02@host ~]$ sudo tail /var/log/secure
[sudo] password for user02: user02_password
user02 is not in the sudoers file. This incident will be reported.
[user02@host ~]$
```

sudo의 또 다른 이점은 기본적으로 실행된 모든 명령을 **/var/log/secure**에 기록한다는 것입니다.

```
[user01@host ~]$ sudo tail /var/log/secure
...output omitted...
Mar 9 20:45:46 host sudo[2577]: user01 : TTY=pts/0 ; PWD=/home/user01 ;
USER=root ; COMMAND=/sbin/usermod -L user02
...output omitted...
```

Red Hat Enterprise Linux 7 이상 버전에서는 **wheel** 그룹의 모든 멤버가 **sudo**를 통해 본인 암호를 사용하여 **root**를 비롯한 임의 사용자로 명령을 실행할 수 있습니다.



경고

지금까지 UNIX 시스템은 **wheel** 그룹의 멤버십을 사용하여 수퍼유저 액세스 권한을 부여하거나 제어합니다. RHEL 6 및 이전 버전에서는 기본적으로 **wheel** 그룹에 어떤 특별한 권한도 부여하지 않습니다. 이전에 이 그룹을 비표준 용도로 사용한 적이 있는 시스템 관리자는 예기치 않은 권한 없는 사용자가 RHEL 7 이상 시스템에서 관리 액세스 권한을 얻지 않도록 이전 구성은 업데이트해야 합니다.

Sudo를 사용하여 대화형 루트 쉘 가져오기

sudo를 사용하여 **root** 계정에 액세스하려면 **sudo -i** 명령을 사용합니다. 이 명령은 **root** 계정으로 전환하고 해당 사용자의 기본 쉘(일반적으로 **bash**) 및 관련된 대화형 로그인 스크립트를 실행합니다. 대화형 스크립트 없이 쉘을 실행하려면 **sudo -s** 명령을 사용합니다.

예를 들어 관리자는 SSH 공개 키 인증을 사용하여 **ec2-user** 일반 사용자로 로그인함으로써 AWS EC2(Elastic Cloud Computing) 인스턴스에서 **root**로 대화형 쉘을 가져올 수 있습니다. 그런 다음 **sudo -i** 명령을 실행하여 **root** 사용자의 쉘에 액세스합니다.

```
[ec2-user@host ~]$ sudo -i
[sudo] password for ec2-user: ec2-user_password
[root@host ~]#
```

sudo 구성

`/etc/sudoers` 파일은 **sudo 명령의 기본 구성 파일입니다**. 여러 관리자가 동시에 파일 편집을 시도하는 경우의 문제를 방지하기 위해 특수 `visudo` 명령을 통해서만 파일을 편집할 수 있습니다. `visudo` 편집기는 파일의 유효성도 검사하여 구문 오류가 없는지 확인합니다.

예를 들어 `/etc/sudoers` 파일의 다음 줄에서는 `wheel` 그룹 멤버에 대해 `sudo` 액세스 권한을 활성화합니다.

```
%wheel      ALL=(ALL:ALL)      ALL
```

- `%wheel` 문자열은 규칙이 적용되는 사용자 또는 그룹입니다. `wheel` 단어 앞의 `%` 기호는 그룹을 지정합니다.
- `ALL=(ALL:ALL)` 명령은 이 파일이 있는 호스트(첫 번째 `ALL`)에서 `wheel` 그룹의 사용자가 시스템에서 다른 사용자(두 번째 `ALL`) 및 다른 그룹(세 번째 `ALL`)으로 명령을 실행할 수 있다고 지정합니다.
- 최종 `ALL` 명령은 `wheel` 그룹의 사용자가 모든 명령을 실행할 수 있다고 지정합니다.

기본적으로 `/etc/sudoers` 파일에는 구성 파일의 일부로 `/etc/sudoers.d` 디렉터리에 있는 모든 파일의 내용도 포함됩니다. 이 계층 구조를 사용하여 해당 디렉터리에 적합한 파일을 배치하면 사용자에 대해 `sudo` 액세스 권한을 추가할 수 있습니다.



참고

파일을 디렉터리에 복사하거나 디렉터리에서 제거하여 `sudo` 액세스 권한을 활성화하거나 비활성화할 수 있습니다.

이 교육 과정에서는 `/etc/sudoers.d` 디렉터리에서 파일을 생성하고 제거하여 사용자 및 그룹에 대해 `sudo` 액세스 권한을 구성합니다.

`user01` 사용자에 대해 전체 `sudo` 액세스 권한을 활성화하려면 다음 내용을 포함하여 `/etc/sudoers.d/user01` 파일을 생성할 수 있습니다.

```
user01      ALL=(ALL)      ALL
```

`group01` 그룹에 대해 전체 `sudo` 액세스 권한을 활성화하려면 다음 내용을 포함하여 `/etc/sudoers.d/group01` 파일을 생성할 수 있습니다.

```
%group01      ALL=(ALL)      ALL
```

`games` 그룹의 사용자가 `operator` 사용자로 `id` 명령을 실행할 수 있도록 하려면 다음 내용을 포함하여 `/etc/sudoers.d/games` 파일을 생성할 수 있습니다.

```
%games  ALL=(operator) /bin/id
```

NOPASSWD: ALL 명령을 사용하면 사용자가 암호를 입력하지 않고 다른 사용자로 명령을 실행할 수 있도록 `sudo`를 설정할 수도 있습니다.

```
ansible      ALL=(ALL)      NOPASSWD: ALL
```

이 액세스 수준을 사용자 또는 그룹에 부여할 경우 명백한 보안 위험이 적용되지만, 시스템 관리자는 서버를 구성하기 위해 클라우드 인스턴스, 가상 시스템 및 프로비저닝 시스템에서 이 방법을 자주 사용합니다. 이 액세스 권한이 있는 계정은 보호해야 하며, 원격 시스템의 사용자가 액세스할 때 SSH 공개 키 인증을 요구해야 합니다.

예를 들어 Amazon Web Services Marketplace의 공식 Red Hat Enterprise Linux용 AMI(Amazon Machine Image)는 **root** 및 **ec2-user** 암호가 잠긴 상태로 제공됩니다. **ec2-user** 계정은 SSH 공개 키 인증을 통한 원격 대화형 액세스를 허용하도록 설정되어 있습니다. AMI의 **/etc/sudoers** 파일에서 마지막 줄이 다음과 같이 설정되었기 때문에 **ec2-user** 사용자가 암호 없이 모든 명령을 **root**로 실행할 수도 있습니다.

```
ec2-user          ALL=(ALL)      NOPASSWD: ALL
```

sudo의 암호를 입력해야 하는 요구 사항을 재활성화하거나, 시스템 구성의 일부로 다른 변경 사항을 도입하여 보안을 강화할 수 있습니다.



참조

su(1), **sudo(8)**, **visudo(8)**, **sudoers(5)** 도움말 페이지

info libc persona(GNU C Library Reference Manual)

- 섹션 30.2: 프로세스의 페르소나

(이 **info** 노드를 사용할 수 있으려면 **glibc-doc** 패키지를 설치해야 합니다.)

▶ 연습 가이드

수퍼유저 액세스 권한 얻기

이 연습에서는 **root** 계정으로 전환하고 **root**로 명령을 실행합니다.

결과

- **sudo** 명령을 사용하여 **root** 사용자로 전환한 다음, 수퍼유저의 암호를 모르고 **root**로 대화형 쉘에 액세스합니다.
- **su** 및 **su -** 명령이 로그인 스크립트를 실행하거나 실행하지 않는 방식으로 쉘 환경에 미치는 영향을 설명합니다.
- **sudo** 명령을 사용하여 **root** 사용자로 다른 명령을 실행합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start users-superuser
```

지침

- ▶ 1. **workstation**에서 **servera** 사용자로 **student**에 대한 SSH 세션을 엽니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **student** 사용자의 쉘 환경을 살펴봅니다. 현재 사용자 및 그룹 정보를 보고 현재 작업 디렉터리를 표시합니다. 또한 사용자의 홈 디렉터리와 사용자 실행 파일의 위치를 지정하는 환경 변수를 봅니다.

2.1. **id**를 실행하여 현재 사용자 및 그룹 정보를 봅니다.

```
[student@servera ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

2.2. **pwd**를 실행하여 현재 작업 디렉터리를 표시합니다.

```
[student@servera ~]$ pwd
/home/student
```

2.3. **HOME** 및 **PATH** 변수의 값을 출력하여 각각 홈 디렉터리와 사용자 실행 파일의 경로를 확인합니다.

```
[student@servera ~]$ echo $HOME
/home/student
[student@servera ~]$ echo $PATH
/home/student/.local/bin:/home/student/bin:/usr/local/bin:/usr/bin:/usr/local/
sbin:/usr/sbin
```

▶ 3. 비로그인 쉘에서 **root** 사용자로 전환하고 새로운 쉘 환경을 살펴봅니다.

- 3.1. 쉘 프롬프트에서 **sudo su** 명령을 실행하여 **root** 사용자가 됩니다.

```
[student@servera ~]$ sudo su
[sudo] password for student: student
[root@servera student]#
```

- 3.2. **id**를 실행하여 현재 사용자 및 그룹 정보를 봅니다.

```
[root@servera student]# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 3.3. **pwd**를 실행하여 현재 작업 디렉터리를 표시합니다.

```
[root@servera student]# pwd
/home/student
```

- 3.4. **HOME** 및 **PATH** 변수의 값을 출력하여 각각 홈 디렉터리와 사용자 실행 파일의 경로를 확인합니다.

```
[root@servera student]# echo $HOME
/root
[root@servera student]# echo $PATH
/root/.local/bin:/root/bin:/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/
local/bin
```

SU 명령을 사용하여 **root** 사용자가 되면 **student** 사용자의 현재 경로가 유지되지 않습니다. 다음 단계에서 볼 수 있듯이, 이 경로도 **root** 사용자 경로가 아닙니다.

결과 차이점은 **SU**를 직접 실행하지 않는다는 것입니다. 대신, 수퍼유저의 암호가 없으므로 **SUDO**를 사용하여 **root** 사용자로 **SU** 명령을 실행합니다. **SUDO** 명령은 보안상의 이유로 환경의 **PATH** 변수를 재정의합니다. 다음 단계에서 볼 수 있듯이, 초기 재정의 후에 실행되는 명령은 **PATH** 변수를 계속 업데이트할 수 있습니다.

- 3.5. **root** 사용자의 쉘을 종료하여 **student** 사용자 쉘로 돌아갑니다.

```
[root@servera student]# exit
exit
[student@servera ~]$
```

▶ 4. 로그인 쉘에서 **root** 사용자로 전환하고 새로운 쉘 환경을 살펴봅니다.

- 4.1. 쉘 프롬프트에서 **sudo su -** 명령을 실행하여 **root** 사용자가 됩니다.

3장 | 로컬 사용자 및 그룹 관리

sudo의 시간 제한 기간에 따라 **sudo** 명령에서 **student** 암호를 확인할 수도 있고, 확인하지 않을 수도 있습니다. 기본 시간 제한 기간은 5분입니다. 최근 5분 이내에 **sudo**에 인증한 경우에는 **sudo** 명령에서 암호를 확인하지 않습니다. **sudo**에 인증한 이후 5분이 지난 경우에는 암호로 **student**를 입력해야 **sudo**에 인증됩니다.

```
[student@servera ~]$ sudo su -
[root@servera ~]#
```

이전 단계의 **sudo su**와 비교해서 쉘 프롬프트의 차이점을 확인합니다.

- 4.2. **id**를 실행하여 현재 사용자 및 그룹 정보를 봅니다.

```
[root@servera ~]# id
uid=0(root) gid=0(root) groups=0(root)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

- 4.3. **pwd**를 실행하여 현재 작업 디렉터리를 표시합니다.

```
[root@servera ~]# pwd
/root
```

- 4.4. **HOME** 및 **PATH** 변수의 값을 출력하여 각각 홈 디렉터리와 사용자 실행 파일의 경로를 확인합니다.

```
[root@servera ~]# echo $HOME
/root
[root@servera ~]# echo $PATH
/root/.local/bin:/root/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin
```

이전 단계와 마찬가지로, **sudo** 명령이 **student** 사용자 쉘 환경의 설정에서 **PATH** 변수를 재설정한 후 **su -** 명령은 **root**에 대해 쉘 로그인 스크립트를 실행하고 **PATH** 변수를 다른 값으로 설정합니다. 대시(-) 옵션이 없는 **su** 명령은 동일한 동작을 수행하지 않습니다.

- 4.5. **root** 사용자의 쉘을 종료하여 **student** 사용자 쉘로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

- ▶ 5. **operator1** 사용자가 **sudo** 명령을 사용하여 임의 사용자로 임의 명령을 실행할 수 있는지 확인합니다.

```
[student@servera ~]$ sudo cat /etc/sudoers.d/operator1
operator1 ALL=(ALL) ALL
```

- ▶ 6. **operator1** 사용자가 되어 **/var/log/messages** 파일의 내용을 봅니다. **/etc/motd** 파일을 **/etc/motdOLD**로 복사합니다. **/etc/motdOLD** 파일을 제거합니다. 이러한 작업에는 관리 권한이 필요하므로 **sudo** 명령을 사용하여 수퍼유저로 명령을 실행합니다. **sudo su** 또는 **sudo su -**를 사용하여 root로 전환하지 마십시오. **redhat** 을 **operator1** 사용자의 암호로 사용합니다.

- 6.1. **operator1** 사용자로 전환합니다.

```
[student@servera ~]$ su - operator1
Password: redhat
[operator1@servera ~]$
```

- 6.2. `sudo`를 사용하지 않고 `/var/log/messages`의 마지막 5줄을 보려고 합니다. 실패하게 됩니다.

```
[operator1@servera ~]$ tail -5 /var/log/messages
tail: cannot open '/var/log/messages' for reading: Permission denied
```

- 6.3. `sudo`를 사용하여 `/var/log/messages`의 마지막 5줄을 보려고 합니다. 성공적으로 실행됩니다.

```
[operator1@servera ~]$ sudo tail -5 /var/log/messages
[sudo] password for operator1: redhat
Mar 9 15:53:36 servera su[2304]: FAILED SU (to operator1) student on pts/1
Mar 9 15:53:51 servera su[2307]: FAILED SU (to operator1) student on pts/1
Mar 9 15:53:58 servera su[2310]: FAILED SU (to operator1) student on pts/1
Mar 9 15:54:12 servera su[2322]: (to operator1) student on pts/1
Mar 9 15:54:25 servera su[2353]: (to operator1) student on pts/1
```



참고

앞의 출력은 시스템에 따라 다를 수 있습니다.

- 6.4. `sudo`를 사용하지 않고 `/etc/motdOLD`로 `/etc/motd`를 복사하려고 합니다. 실패하게 됩니다.

```
[operator1@servera ~]$ cp /etc/motd /etc/motdOLD
cp: cannot create regular file '/etc/motdOLD': Permission denied
```

- 6.5. `sudo`를 사용하여 `/etc/motdOLD`로 `/etc/motd`를 복사하려고 합니다. 성공적으로 실행됩니다.

```
[operator1@servera ~]$ sudo cp /etc/motd /etc/motdOLD
[operator1@servera ~]$
```

- 6.6. `sudo`를 사용하지 않고 `/etc/motdOLD`를 삭제하려고 합니다. 실패하게 됩니다.

```
[operator1@servera ~]$ rm /etc/motdOLD
rm: remove write-protected regular empty file '/etc/motdOLD'? y
rm: cannot remove '/etc/motdOLD': Permission denied
[operator1@servera ~]$
```

- 6.7. `sudo`를 사용하여 `/etc/motdOLD`를 삭제하려고 합니다. 성공적으로 실행됩니다.

```
[operator1@servera ~]$ sudo rm /etc/motdOLD
[operator1@servera ~]$
```

6.8. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[operator1@servera ~]$ exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-superuser
```

이것으로 섹션을 완료합니다.

로컬 사용자 계정 관리

목표

로컬 사용자 계정을 생성, 수정 및 삭제합니다.

로컬 사용자 관리

명령줄 툴을 사용하여 로컬 사용자 계정을 관리할 수 있습니다. 이 섹션에서는 중요한 몇 가지 툴을 검토합니다.

명령줄에서 사용자 생성

`useradd username` 명령은 `username`이라는 사용자를 생성합니다. 사용자의 홈 디렉터리 및 계정 정보를 설정하고 `username`이라는 사용자의 개인 그룹을 생성합니다. 이 시점에는 계정에 유효한 암호가 설정되어 있지 않으며, 사용자는 암호가 설정될 때까지 로그인할 수 없습니다.

`useradd --help` 명령은 기본값을 재정의하는 기본 옵션을 표시합니다. 대부분의 경우 `usermod` 명령에 동일한 옵션을 사용하여 기존 사용자를 수정할 수 있습니다.

`/etc/login.defs` 파일은 유효한 UID 번호 범위 및 기본 암호 에이징 규칙과 같은 몇 가지 기본 옵션을 사용자 계정에 대해 설정합니다. 이 파일의 값은 새롭게 생성한 사용자 계정에만 적용됩니다. 이 파일의 변경 사항은 기존 사용자에게 영향을 주지 않습니다.

Red Hat Enterprise Linux 9에서 `useradd` 명령은 `-u` 옵션을 사용하여 명시적으로 UID를 지정하지 않는 한, 1000보다 크거나 같은 사용 가능한 첫 번째 UID를 새 사용자에게 할당합니다.

명령줄에서 기존 사용자 수정

`usermod --help` 명령은 계정을 수정하는 옵션을 표시합니다. 몇 가지 일반적인 옵션은 다음과 같습니다.

usermod 옵션:	사용법
<code>-a, --append</code>	보조 그룹 집합을 새로운 집합으로 바꾸는 대신 보조 그룹을 사용자의 현재 그룹 멤버십 집합에 추가 하려면 <code>-G</code> 옵션과 함께 사용합니다.
<code>-c, --comment COMMENT</code>	주석 필드에 <code>COMMENT</code> 텍스트를 추가합니다.
<code>-d, --home HOME_DIR</code>	사용자 계정의 홈 디렉터리를 지정합니다.
<code>-g, --gid GROUP</code>	사용자 계정에 대한 기본 그룹을 지정합니다.
<code>-G, --groups GROUPS</code>	사용자 계정에 대해 쉼표로 구분된 보조 그룹 목록을 지정합니다.
<code>-L, --lock</code>	사용자 계정을 잠금니다.
<code>-m, --move-home</code>	사용자의 홈 디렉터리를 새 위치로 이동합니다. <code>-d</code> 옵션과 함께 사용해야 합니다.

usermod 옵션:	사용법
-S, --shell SHELL	사용자 계정에 대한 특정 로그인 쉘을 지정합니다.
-U, --unlock	사용자 계정의 잠금을 해제합니다.

명령줄에서 사용자 삭제

`userdel username` 명령은 `/etc/passwd`에서 `username` 사용자를 제거하지만 사용자의 홈 디렉터리는 그대로 유지됩니다. `userdel -r username` 명령은 `/etc/passwd`에서 사용자를 제거하고 사용자의 홈 디렉터리를 삭제합니다.



경고

`userdel -r` 옵션을 지정하지 않고 사용자를 제거하면 이제 할당되지 않은 UID가 사용자 파일을 소유합니다. 사용자를 생성하고 해당 사용자에게 삭제된 사용자의 UID가 할당되면 새 계정이 해당 파일을 소유하므로 보안 위험이 있습니다. 일반적으로 조직 보안 정책은 이 시나리오를 방지하기 위해 사용자 계정의 삭제를 허용하지 않으며, 대신 사용하지 못하도록 계정을 잠깁니다.

다음 예제에서는 이 시나리오에서 어떻게 정보 유출이 발생할 수 있는지를 보여줍니다.

```
[root@host ~]# useradd user01
[root@host ~]# ls -l /home
drwx----- 3 user01 user01 74 Mar 4 15:22 user01
[root@host ~]# userdel user01
[root@host ~]# ls -l /home
drwx----- 3 1000 1000 74 Mar 4 15:22 user01
[root@host ~]# useradd -u 1000 user02
[root@host ~]# ls -l /home
drwx----- 3 user02 user02 74 Mar 4 15:23 user02
drwx----- 3 user02 user02 74 Mar 4 15:22 user01
```

`user02`이 이전에 소유했던 모든 파일을 이제 `user01`가 소유합니다. `root` 사용자는 `find / -nouser -o -nogroup` 명령을 사용하여 소유되지 않은 모든 파일 및 디렉터리를 찾을 수 있습니다.

명령줄에서 암호 설정

`passwd username` 명령은 `username` 사용자의 초기 암호를 설정하거나 기존 암호를 변경합니다. `root` 사용자는 암호를 어떤 값으로든 설정할 수 있습니다. 암호가 최소 권장 기준에 맞지 않으면 터미널에 메시지가 표시되지만 새 암호를 다시 입력할 수 있으며 `passwd` 명령이 성공적으로 암호를 업데이트합니다.

```
[root@host ~]# passwd user01
Changing password for user user01.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@host ~]#
```

일반 사용자는 8자 이상의 암호를 선택해야 합니다. 사전 단어, 사용자 이름 또는 이전 암호는 사용하지 마십시오.

UID 범위

Red Hat Enterprise Linux에서 특정 UID 번호와 번호 범위는 특정 목적으로 사용됩니다.

- **UID 0** : 수퍼유저(**root**) 계정 UID입니다.
- **UID 1~200** : 시스템 프로세스에 정적으로 할당되는 시스템 계정 UID입니다.
- **UID 201~999** : 이 시스템의 파일을 소유하지 않은 시스템 프로세스에 할당되는 UID입니다. 권한 없는 UID가 필요한 소프트웨어에는 이 사용 가능한 풀의 UID가 동적으로 할당됩니다.
- **UID 1000 이상** : 권한 없는 일반 사용자에게 할당할 UID 범위입니다.



참고

RHEL 6 및 이전 버전에서는 시스템 사용자에 1~499 범위의 UID를 사용하고, 일반 사용자에 500 이상의 UID를 사용합니다. `/etc/login.defs` 파일에서 **useradd** 및 **groupadd** 기본 범위를 변경할 수 있습니다.



참조

useradd(8), **usermod(8)** 및 **userdel(8)** 도움말 페이지

▶ 연습 가이드

로컬 사용자 계정 관리

이 연습에서는 시스템에 여러 사용자를 생성하고 해당 사용자의 암호를 설정합니다.

결과

- 추가 사용자 계정을 사용하여 Linux 시스템을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start users-user
```

지침

- ▶ 1. **workstation**에서 **student** 사용자로 **servera**에 대한 SSH 세션을 열고 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **operator1** 사용자를 생성하고 시스템에 있는지 확인합니다.

```
[root@servera ~]# useradd operator1
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002::/home/operator1:/bin/bash
```

- ▶ 3. **operator1**의 암호를 **redhat**으로 설정합니다.

```
[root@servera ~]# passwd operator1
Changing password for user operator1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- ▶ 4. **operator2** 및 **operator3** 사용자를 추가로 생성합니다. 사용자 암호를 **redhat**으로 설정합니다.

- 4.1. operator2 사용자를 추가합니다. operator2의 암호를 redhat으로 설정합니다.

```
[root@servera ~]# useradd operator2
[root@servera ~]# passwd operator2
Changing password for user operator2.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 4.2. operator3 사용자를 추가합니다. operator3의 암호를 redhat으로 설정합니다.

```
[root@servera ~]# useradd operator3
[root@servera ~]# passwd operator3
Changing password for user operator3.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- ▶ 5. operator1 및 operator2 사용자 계정에 각각 Operator One 및 Operator Two 주석이 포함되도록 업데이트합니다. 사용자 계정에 대한 주석이 있는지 확인합니다.

- 5.1. usermod -c 명령을 실행하여 operator1 사용자 계정의 주석을 업데이트합니다.

```
[root@servera ~]# usermod -c "Operator One" operator1
```

- 5.2. usermod -c 명령을 실행하여 operator2 사용자 계정의 주석을 업데이트합니다.

```
[root@servera ~]# usermod -c "Operator Two" operator2
```

- 5.3. /etc/passwd 파일을 보고 operator1 및 operator2 사용자에 대한 주석이 있는지 확인합니다.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
operator3:x:1004:1004::/home/operator3:/bin/bash
```

- ▶ 6. 사용자의 개인 데이터와 함께 operator3 사용자를 삭제합니다. operator3이 없는지 확인합니다.

- 6.1. 시스템에서 operator3 사용자를 제거합니다.

```
[root@servera ~]# userdel -r operator3
```

- 6.2. operator3 사용자가 없는지 확인합니다.

```
[root@servera ~]# tail /etc/passwd
...output omitted...
operator1:x:1002:1002:Operator One:/home/operator1:/bin/bash
operator2:x:1003:1003:Operator Two:/home/operator2:/bin/bash
```

앞의 출력에는 **operator3**의 사용자 계정 정보가 표시되지 않습니다.

- 6.3. **operator3** 사용자 홈 디렉터리가 없는지 확인합니다.

```
[root@servera ~]# ls -l /home
total 0
drwx----- 4 devops    devops    90 Mar  3 09:59 devops
drwx----- 2 operator1 operator1 62 Mar  9 10:19 operator1
drwx----- 2 operator2 operator2 62 Mar  9 10:19 operator2
drwx----- 3 student   student   95 Mar  3 09:49 student
```

- 6.4. **root** 사용자의 쉘을 종료하여 **student** 사용자 쉘로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$
```

- 6.5. **servera** 시스템에서 로그아웃합니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-user
```

이것으로 섹션을 완료합니다.

로컬 그룹 계정 관리

목표

로컬 그룹 계정을 생성, 수정 및 삭제합니다.

로컬 그룹 관리

여러 명령줄 툴이 그룹 관리를 지원합니다. **Users** GUI 유ти리티를 사용하여 그룹을 관리할 수 있지만 Red Hat에서는 명령줄 툴을 사용하도록 권장합니다.

명령줄에서 그룹 생성

groupadd 명령은 그룹을 생성합니다. 옵션이 지정되지 않은 **groupadd** 명령은 **/etc/login.defs** 파일에서 **GID_MIN** 및 **GID_MAX** 변수가 지정하는 범위에서 사용 가능한 다음 GID를 사용합니다. 기본적으로 이 명령은 더 낮은 값을 사용할 수 있게 되더라도 다른 기존 GID보다 큰 GID 값을 할당합니다.

groupadd 명령의 **-g** 옵션은 그룹에 사용할 GID를 지정합니다.

```
[root@host ~]# groupadd -g 10000 group01
[root@host ~]# tail /etc/group
...output omitted...
group01:x:10000:
```



참고

사용자 개인 그룹(GID 1000 이상)이 자동 생성되기 때문에 일부 관리자는 다른 용도로 보조 그룹을 생성하기 위해 별도의 GID 범위를 따로 관리합니다. 그러나 사용자의 UID와 기본 GID 가 같은 번호일 필요는 없으므로 이 추가 관리는 불필요합니다.

groupadd 명령의 **-r** 옵션은 시스템 그룹을 생성합니다. 일반 그룹과 마찬가지로, 시스템 그룹은 **/etc/login.defs** 파일에 표시된 유효한 시스템 GID 범위의 GID를 사용합니다. **/etc/login.defs** 파일의 **SYS_GID_MIN** 및 **SYS_GID_MAX** 구성 항목이 시스템 GID 범위를 정의합니다.

```
[root@host ~]# groupadd -r group02
[root@host ~]# tail /etc/group
...output omitted...
group01:x:10000:
group02:x:988:
```

명령줄에서 기존 그룹 수정

groupmod 명령은 기존 그룹의 등록 정보를 변경합니다. **groupmod** 명령의 **-n** 옵션은 그룹의 새 이름을 지정합니다.

```
[root@host ~]# groupmod -n group0022 group02
[root@host ~]# tail /etc/group
...output omitted...
group0022:x:988:
```

그룹 이름이 group02에서 group0022로 업데이트됩니다. **groupmod** 명령의 **-g** 옵션은 새 GID를 지정합니다.

```
[root@host ~]# groupmod -g 20000 group0022
[root@host ~]# tail /etc/group
...output omitted...
group0022:x:20000:
```

GID가 988에서 20000으로 변경됩니다.

명령줄에서 그룹 삭제

groupdel 명령은 그룹을 제거합니다.

```
[root@host ~]# groupdel group0022
```



참고

기존 사용자의 기본 그룹인 경우에는 그룹을 제거할 수 없습니다. **userdel** 명령을 사용하는 것과 유사하게, 먼저 그룹이 소유한 파일을 찾아야 합니다.

명령줄에서 그룹 멤버십 변경

그룹 구성원은 사용자 관리로 제어됩니다. **usermod -g** 명령을 사용하여 사용자의 기본 그룹을 변경합니다.

```
[root@host ~]# id user02
uid=1006(user02) gid=1008(user02) groups=1008(user02)
[root@host ~]# usermod -g group01 user02
[root@host ~]# id user02
uid=1006(user02) gid=10000(group01) groups=10000(group01)
```

usermod -aG 명령을 사용하여 사용자를 보조 그룹에 추가합니다.

```
[root@host ~]# id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03)
[root@host ~]# usermod -aG group01 user03
[root@host ~]# id user03
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
```



중요

usermod 명령의 **-a** 옵션은 추가 모드를 활성화합니다. **-a** 옵션이 지정되지 않은 경우 이 명령은 **-G** 옵션 목록에 포함되지 않은 모든 현재 보조 그룹에서 사용자를 제거합니다.

기본 및 보조 그룹 멤버십 비교

사용자의 기본 그룹은 **/etc/passwd** 파일에서 사용자 계정에 표시되는 그룹입니다. 사용자는 한 번에 하나의 기본 그룹에만 속할 수 있습니다.

사용자의 보조 그룹은 사용자에 대해 구성되고 **/etc/group** 파일에서 사용자 항목에 표시되는 추가 그룹입니다. 사용자는 파일 액세스 및 권한을 효과적으로 구현하는데 필요한 개수만큼 많은 보조 그룹에 속할 수 있습니다.

그룹 기반 파일 권한을 구성할 때는 사용자의 기본 그룹과 보조 그룹 간에 차이가 없습니다. 특정 파일에 대한 액세스 권한이 할당된 그룹에 속해 있는 사용자는 해당 파일에 액세스할 수 있습니다.

사용자의 기본 멤버십과 보조 멤버십에는 사용자가 파일을 생성할 때만 차이가 있습니다. 새 파일에는 파일이 생성될 때 할당되는 사용자 소유자와 그룹 소유자가 있어야 합니다. 명령 옵션이 재정의하지 않는 한, 사용자의 기본 그룹이 새 파일의 그룹 소유권에 사용됩니다.

일시적으로 기본 그룹 변경

사용자의 기본 그룹만 새 파일 생성 속성에 사용됩니다. 그러나 일시적으로 기본 그룹을 이미 속해 있는 보조 그룹으로 전환할 수 있습니다. 파일을 수동으로 또는 스크립팅하여 생성하려고 하며, 생성 시 다른 그룹을 소유자로 할당하려는 경우에 전환할 수 있습니다.

newgrp 명령을 사용하여 이 쉘 세션에서 기본 그룹을 전환합니다. 속해 있는 모든 기본 그룹 또는 보조 그룹 간에 전환할 수 있지만, 한 번에 한 그룹만 기본 그룹이 될 수 있습니다. 로그아웃한 후 다시 로그인하면 기본 그룹이 기본값으로 돌아갑니다. 이 예제에서는 **group01** 그룹이 일시적으로 사용자의 기본 그룹이 됩니다.

```
[user03@host ~]# id
uid=1007(user03) gid=1009(user03) groups=1009(user03),10000(group01)
[user03@host ~]$ newgrp group01
[user03@host ~]# id
uid=1007(user03) gid=10000(group01) groups=1009(user03),10000(group01)
```



참조

group(5), **groupadd(8)**, **groupdel(8)**, **usermod(8)** 도움말 페이지

▶ 연습 가이드

로컬 그룹 계정 관리

이 연습에서는 그룹을 생성하고, 사용자의 기본 그룹을 변경하지 않고 해당 그룹을 일부 사용자의 보조 그룹으로 사용하고, **root**로 명령을 실행할 수 있는 **sudo** 액세스 권한으로 그룹 중 하나를 구성합니다.

결과

- 그룹을 생성하여 보조 그룹으로 사용합니다.
- 그룹에 대해 **sudo** 액세스 권한을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 올바르게 설정하는데 필요한 사용자 계정을 생성합니다.

```
[student@workstation ~]$ lab start users-group
```

지침

- ▶ 1. **workstation**에서 **student** 사용자로 **servera**에 대한 SSH 세션을 열고 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. GID가 30000인 **operators** 보조 그룹을 생성합니다.

```
[root@servera ~]# groupadd -g 30000 operators
```

- ▶ 3. GID를 지정하지 않고 **admin** 보조 그룹을 생성합니다.

```
[root@servera ~]# groupadd admin
```

- ▶ 4. **operators** 및 **admin** 보조 그룹이 있는지 확인합니다.

```
[root@servera ~]# tail /etc/group
...output omitted...
operators:x:30000:
admin:x:30001:
```

▶ 5. operator1, operator2, operator3 사용자가 operators 그룹에 속하게 합니다.

- 5.1. operator1, operator2, operator3 사용자를 operators 그룹에 추가합니다.

```
[root@servera ~]# usermod -aG operators operator1
[root@servera ~]# usermod -aG operators operator2
[root@servera ~]# usermod -aG operators operator3
```

- 5.2. 사용자가 그룹에 속해 있는지 확인합니다.

```
[root@servera ~]# id operator1
uid=1002(operator1) gid=1002(operator1) groups=1002(operator1),30000(operators)
[root@servera ~]# id operator2
uid=1003(operator2) gid=1003(operator2) groups=1003(operator2),30000(operators)
[root@servera ~]# id operator3
uid=1004(operator3) gid=1004(operator3) groups=1004(operator3),30000(operators)
```

▶ 6. sysadmin1, sysadmin2, sysadmin3 사용자가 admin 그룹에 속하게 합니다. 모든 admin 그룹 멤버에 대해 관리 권한을 활성화합니다. admin 그룹의 모든 멤버가 관리 명령을 실행할 수 있는지 확인합니다.

- 6.1. sysadmin1, sysadmin2, sysadmin3 사용자를 admin 그룹에 추가합니다.

```
[root@servera ~]# usermod -aG admin sysadmin1
[root@servera ~]# usermod -aG admin sysadmin2
[root@servera ~]# usermod -aG admin sysadmin3
```

- 6.2. 사용자가 그룹에 속해 있는지 확인합니다.

```
[root@servera ~]# id sysadmin1
uid=1005(sysadmin1) gid=1005(sysadmin1) groups=1005(sysadmin1),30001(admin)
[root@servera ~]# id sysadmin2
uid=1006(sysadmin2) gid=1006(sysadmin2) groups=1006(sysadmin2),30001(admin)
[root@servera ~]# id sysadmin3
uid=1007(sysadmin3) gid=1007(sysadmin3) groups=1007(sysadmin3),30001(admin)
```

- 6.3. /etc/group 파일을 검사하여 보조 그룹 멤버십을 확인합니다.

```
[root@servera ~]# tail /etc/group
...output omitted...
operators:x:30000:operator1,operator2,operator3
admin:x:30001:sysadmin1,sysadmin2,sysadmin3
```

- 6.4. admin 그룹의 멤버가 전체 관리 권한을 갖도록 /etc/sudoers.d/admin 파일을 생성합니다.

```
[root@servera ~]# echo "%admin ALL=(ALL) ALL" >> /etc/sudoers.d/admin
```

- 6.5. **sysadmin1** 사용자(admin 그룹의 멤버)로 전환하고, **sudo** 명령을 실행할 수 있는지 확인합니다.

```
[root@servera ~]# su - sysadmin1
[sysadmin1@servera ~]$ sudo cat /etc/sudoers.d/admin
[sudo] password for sysadmin1: redhat
%admin ALL=(ALL) ALL
```

- 6.6. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[sysadmin1@servera ~]$ exit
logout
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-group
```

이것으로 섹션을 완료합니다.

사용자 암호 관리

목표

사용자에 대한 암호 관리 정책을 설정하고, 수동으로 사용자 계정을 잠그거나 잠금을 해제합니다.

섀도 암호 및 암호 정책

암호화된 암호는 원래 전역적으로 읽기 가능한 `/etc/passwd` 파일에 저장되었습니다. 이런 암호는 암호화된 암호에 대한 사전 공격이 일반화될 때까지는 적절한 방법으로 간주되었습니다. 암호화 방식의 해시 암호가 `root` 사용자만 읽을 수 있는 `/etc/shadow` 파일로 이동되었습니다.

`/etc/passwd` 파일과 마찬가지로, `/etc/shadow` 파일에는 각 사용자의 항목이 있습니다. `/etc/shadow` 파일의 예제 항목에는 콜론으로 구분된 필드 9개가 있습니다.

```
[root@host ~]# cat /etc/shadow
...output omitted...
user03:$6$C5sXsd3rwghsdfarf:17933:0:99999:7:2:18113:
```

이 코드 블록의 각 필드는 콜론으로 구분되어 있습니다.

- **user03** : 사용자 계정의 이름입니다.
- **\$6\$C5sXsd3rwghsdfarf** : 암호화 방식의 해시 사용자 암호입니다.
- **17933** : epoch부터 암호가 마지막으로 변경된 시점까지의 일수입니다. 여기서 epoch는 **1970-01-01** (UTC 시간대)입니다.
- **0**: 사용자가 다시 변경할 수 있기 전에 마지막 암호가 변경된 이후 경과해야 하는 최소 일수입니다.
- **99999**: 암호가 만료되기 전에 암호 변경 없이 유지할 수 있는 최대 일수입니다. 빈 필드는 암호가 만료되지 않음을 의미합니다.
- **7** : 사용자에게 암호 만료를 미리 경고하는 일수입니다.
- **2** : 암호가 만료된 날부터 계정이 자동으로 잠기기 전에 활동 없이 유지할 수 있는 일수입니다.
- **18113** : epoch 이후 계정이 만료되는 날까지의 일수입니다. 빈 필드는 계정이 만료되지 않음을 의미합니다.
- 마지막 필드는 일반적으로 비어 있으며, 나중에 사용하기 위해 예약되었습니다.

암호화 방식의 해시 암호 형식

암호화된 방식의 해시 암호 필드에는 사용 중인 해시 알고리즘, salt, 암호화 방식의 해시라는 세 가지 정보가 저장됩니다. Salt는 암호화된 방식의 해시에 임의의 데이터를 추가하여 고유한 해시를 생성하여 암호화된 방식의 해시 암호를 강화합니다. 각 정보는 달러(\$) 문자로 구분됩니다.

```
$6$C5sXcYG1L/4ZfHr/$2W6evvJahUfzfHpc9X.45Jc6H30E
```

- **6** : 이 암호에 사용 중인 해시 알고리즘입니다. **6**은 RHEL 9 기본값인 SHA-512 해시를 나타내고, **1**은 MD5를 나타내고, **5**는 SHA-256을 나타냅니다.

- C\$eXcYG1L/4ZfHr/** : 암호를 암호화된 방식의 해시로 만들 때 사용하는 salt입니다. 원래 임의로 선택되었습니다.
- 2W6evvJahUfzfHpc9X.45Jc6H30E** : salt와 일반 텍스트 암호를 결합한 후 암호화된 방식의 암호 해시를 생성하는 사용자 암호의 암호화된 방식의 해시입니다.

salt를 암호와 조합하는 주된 이유는 미리 계산된 암호 해시 목록을 사용하는 공격으로부터 보호하기 위한 것입니다. salt를 추가하면 결과 해시가 변경되므로 미리 계산된 목록이 쓸모없게 됩니다. 공격자가 salt를 사용하는 **/etc/shadow** 파일의 복사본을 얻으면 무차별 암호 대입을 통해 암호를 추측해야 하므로 더 많은 시간과 노력이 필요합니다.

암호 확인

사용자가 로그인을 시도하면 시스템은 **/etc/shadow** 파일에서 그 사용자에 해당하는 항목을 찾고, 일반 텍스트 암호를 사용하여 사용자의 salt를 결합합니다. 그런 다음 시스템은 지정된 해시 알고리즘을 사용하여 salt 및 일반 텍스트 암호의 조합을 암호화된 방식으로 해시합니다. 결과가 암호화된 방식의 해시와 일치하면 사용자가 올바른 암호를 입력한 것입니다. 결과가 암호화된 방식의 해시와 일치하지 않으면 사용자가 잘못된 암호를 입력한 것이며 로그인 시도가 실패합니다. 이 방법을 사용하면 시스템에서 로그인에 사용 가능한 형식으로 암호를 저장하지 않고 사용자가 올바른 암호를 입력했는지 여부를 확인할 수 있습니다.

암호 에이징 구성

다음 다이어그램은 암호 에이징 정책을 구현하기 위해 **chage** 명령을 사용하여 조정할 수 있는 관련 암호 에이징 매개 변수를 보여줍니다. 명령 이름은 "change age"를 나타내는 **chage**입니다. 명령을 "change" 단어와 혼동하지 마십시오.

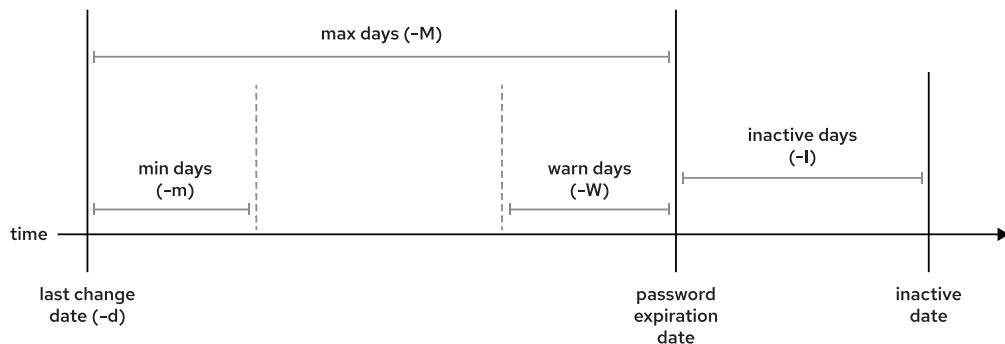


그림 3.1: 암호 사용 기간 매개 변수

다음 예제에서는 **sysadmin05** 사용자의 암호 정책을 변경하는 **chage** 명령을 보여줍니다. 이 명령은 최소 사용 기간(**-m**)을 0일, 최대 사용 기간(**-M**)을 90일, 경고 기간(**-W**)을 7일, 비활성 기간(**-I**)을 14일로 정의합니다.

```
[root@host ~]# chage -m 0 -M 90 -W 7 -I 14 sysadmin05
```

Red Hat 서버에서 사용자 암호 정책을 관리한다고 가정합니다. **cloudadmin10** 사용자는 시스템의 새 사용자이며, 사용자 지정 암호 에이징 정책을 설정하려고 합니다. 오늘부터 30일 후에 계정이 만료되도록 설정하려고 하므로 다음 명령을 사용합니다.

```
[root@host ~]# date +%F ①
2022-03-10
[root@host ~]# date -d "+30 days" +%F ②
2022-04-09
[root@host ~]# chage -E $(date -d "+30 days" +%F) cloudadmin10 ③
[root@host ~]# chage -l cloudadmin10 | grep "Account expires" ④
Account expires      : Apr 09, 2022
```

- ① `date` 명령을 사용하여 현재 날짜를 가져옵니다.
- ② `date` 명령을 사용하여 지금부터 30일 후의 날짜를 가져옵니다.
- ③ `chage` 명령의 `-E` 옵션을 사용하여 `cloudadmin10` 사용자의 만료 날짜를 변경합니다.
- ④ `chage` 명령의 `-l` 옵션을 사용하여 `cloudadmin10` 사용자의 암호 에이징 정책을 표시합니다.

며칠 후 `/var/log/secure` 로그 파일에서 `cloudadmin10` 사용자가 비정상적으로 동작했음을 확인합니다. 사용자가 `sudo` 를 사용하여 다른 사용자에게 속한 파일을 조작하려고 했습니다. 사용자가 다른 시스템에서 작업하는 동안 `ssh` 세션을 열어 두었을 수 있다고 의심합니다. `cloudadmin10` 사용자가 다음 로그인 시 암호를 변경하기를 원하므로 다음 명령을 사용합니다.

```
[root@host ~]# chage -d 0 cloudadmin10
```

다음에 `cloudadmin10` 사용자가 로그인하면 암호를 변경하라는 메시지가 표시됩니다.



참고

`date` 명령은 이후 날짜를 계산할 수 있습니다. `-u` 옵션은 UTC로 시간을 보고합니다.

```
[user01@host ~]$ date -d "+45 days" -u
Thu May 23 17:01:20 UTC 2019
```

`/etc/login.defs` 파일에서 기본 암호 에이징 구성을 변경할 수 있습니다. `PASS_MAX_DAYS` 및 `PASS_MIN_DAYS` 옵션은 각각 암호의 기본 최대 및 최소 사용 기간을 설정합니다. `PASS_WARN_AGE`는 암호의 기본 경고 기간을 설정합니다. 기본 암호 에이징 정책을 변경하면 변경 후 생성되는 사용자에게 영향을 줍니다. 기존 사용자는 이후 암호 에이징 설정보다 오래된 암호 에이징 설정을 계속 사용합니다. `/etc/login.defs` 파일에 대한 자세한 내용은 Red Hat Security: Linux in Physical, Virtual, and Cloud (RH415) 교육 과정 및 `login.defs(5)` 도움말 페이지를 참조하십시오.

액세스 제한

`usermod` 명령을 사용하여 사용자의 계정 만료를 수정할 수 있습니다. 예를 들어 `usermod` 명령의 `-L` 옵션은 사용자 계정을 잠그므로 사용자가 시스템에 로그인할 수 없습니다.

```
[root@host ~]# usermod -L sysadmin03
[user01@host ~]$ su - sysadmin03
Password: redhat
su: Authentication failure
```

3장 | 로컬 사용자 및 그룹 관리

사용자가 특정 날짜에 퇴사하는 경우 단일 **usermod** 명령을 사용하여 계정을 잠그고 만료할 수 있습니다. 날짜는 **1970-01-01** 이후 경과한 일수이거나 YYYY-MM-DD 형식을 사용해야 합니다. 다음 예제에서 **usermod** 명령은 2022-08-14에 **cloudadmin10** 사용자를 잠그고 만료합니다.

```
[root@host ~]# usermod -L -e 2022-08-14 cloudadmin10
```

계정을 잠그면 사용자가 암호를 사용하여 시스템에 인증할 수 없게 됩니다. 이 방법은 회사의 이전 직원이 계정에 액세스하지 못하도록 차단하는 데 권장됩니다. **usermod** 명령의 **-U** 옵션을 사용하면 계정에 대한 액세스를 다시 활성화할 수 있습니다.

nologin 쉘

nologin 쉘은 대화형으로 시스템에 로그인하지 않으려는 사용자 계정을 위한 대체 쉘 역할을 합니다. 계정에 필요하지 않은 경우 시스템에 로그인할 수 없도록 계정을 비활성화하는 것이 보안상 좋습니다. 예를 들어 메일 서버는 메일을 저장할 계정과 메일 검색을 위해 메일 클라이언트에 인증하는 사용자의 암호를 요구할 수 있습니다. 해당 사용자가 시스템에 직접 로그인할 필요는 없습니다.

이 경우의 일반적인 해결책은 사용자의 로그인 쉘을 **/sbin/nologin**으로 설정하는 것입니다. 사용자가 시스템에 직접 로그인하려고 하면 **nologin** 쉘이 연결을 닫습니다.

```
[root@host ~]# usermod -s /sbin/nologin newapp  
[root@host ~]# su - newapp  
Last login: Wed Feb 6 17:03:06 IST 2019 on pts/0  
This account is currently not available.
```



중요

nologin 쉘은 시스템에 대한 대화형 접근을 차단하지만 모든 액세스를 차단하는 것은 아닙니다. 인증을 위해 사용자의 암호를 사용하는 경우 사용자는 웹 애플리케이션, 파일 전송 프로그램, 메일 리더 등의 애플리케이션을 통해 계속 파일을 인증 및 업로드하거나 검색할 수 있습니다.



참조

chage(1), **usermod(8)**, **shadow(5)**, **crypt(3)**, **login.defs(5)** 도움말 페이지

▶ 연습 가이드

사용자 암호 관리

이 연습에서는 여러 사용자의 암호 정책을 설정합니다.

결과

- 사용자가 시스템에 처음 로그인할 때 강제로 암호를 변경합니다.
- 90일마다 강제로 암호를 변경합니다.
- 현재 날짜부터 180일 후에 만료되도록 계정을 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start users-password
```

지침

- ▶ 1. **workstation**에서 **student** 사용자로 **servera** 시스템에 대한 SSH 세션을 엽니다.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$
```

- ▶ 2. **servera**에서 **usermod** 명령을 사용하여 **operator1** 사용자를 잠그고 잠금을 해제합니다.

- 2.1. **student** 사용자로 관리 권한을 사용하여 **operator1** 계정을 잠급니다.

```
[student@servera ~]$ sudo usermod -L operator1
[sudo] password for student: student
```

- 2.2. **operator1**로 로그인을 시도합니다. 이 명령은 실패합니다.

```
[student@servera ~]$ su - operator1
Password: redhat
su: Authentication failure
```

- 2.3. **operator1** 계정을 잠금 해제합니다.

```
[student@servera ~]$ sudo usermod -U operator1
```

- 2.4. 다시 **operator1**로 로그인을 시도합니다. 이번에는 명령이 성공합니다.

```
[student@servera ~]$ su - operator1  
Password: redhat  
...output omitted...  
[operator1@servera ~]$
```

2.5. **operator1** 사용자 쉘에서 로그아웃하여 **student** 사용자 쉘로 돌아갑니다.

```
[operator1@servera ~]$ exit  
logout
```

- ▶ 3. 90일마다 새 암호를 요구하도록 **operator1** 사용자의 암호 정책을 변경합니다. 암호 사용 기간이 성공적으로 설정되었는지 확인합니다.

3.1. **root** 사용자로 전환합니다.

```
[student@servera ~]$ sudo -i  
[sudo] password for student: student  
[root@servera ~]#
```

3.2. **operator1** 사용자 암호의 최대 사용 기간을 90일로 설정합니다.

```
[root@servera ~]# chage -M 90 operator1
```

3.3. **operator1** 사용자의 암호가 변경된 날부터 90일 후에 만료되는지 확인합니다.

```
[root@servera ~]# chage -l operator1  
Last password change      : Mar 10, 2022  
Password expires          : Jun 10, 2022  
Password inactive         : never  
Account expires           : never  
Minimum number of days between password change   : 0  
Maximum number of days between password change   : 90  
Number of days of warning before password expires : 7
```

- ▶ 4. **operator1** 계정으로 처음 로그인할 때 암호를 강제로 변경합니다.

```
[root@servera ~]# chage -d 0 operator1
```

- ▶ 5. **servera** 시스템에서 **root** 사용자를 종료합니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$
```

- ▶ 6. **operator1**로 로그인하고 암호를 **forsooth123**으로 변경합니다. 암호를 설정한 후 **student** 사용자 쉘로 돌아갑니다.

6.1. **operator1**로 로그인하고, 메시지가 표시되면 암호를 **forsooth123**으로 변경합니다.

```
[student@servera ~]$ su - operator1
Password: redhat
You are required to change your password immediately (administrator enforced)
Current password: redhat
New password: forsooth123
Retype new password: forsooth123
...output omitted...
[operator1@servera ~]$
```

- 6.2. **operator1** 사용자의 쉘을 종료하여 **student** 사용자로 돌아간 다음, **root** 사용자로 전환합니다.

```
[operator1@servera ~]$ exit
logout
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

▶ 7. 현재 날짜로부터 180일 후에 만료되도록 **operator1** 계정을 설정합니다.

- 7.1. 180일 후가 될 날짜를 결정합니다. 정확한 값을 얻으려면 %F 명령에 **date** 형식을 사용합니다. 이 반환된 날짜는 하나의 예입니다. 이후 단계에서는 해당 시스템의 값을 사용합니다.

```
[root@servera ~]# date -d "+180 days" +%F
2022-09-06
```

- 7.2. 앞의 단계에서 표시된 날짜에 만료되도록 계정을 설정합니다. 예를 들면 다음과 같습니다.

```
[root@servera ~]# chage -E 2022-09-06 operator1
```

- 7.3. 계정 만료일이 성공적으로 설정되었는지 확인합니다.

```
[root@servera ~]# chage -l operator1
Last password change      : Mar 10, 2022
Password expires          : Jun 10, 2022
Password inactive         : never
Account expires           : Sep 06, 2022
Minimum number of days between password change   : 0
Maximum number of days between password change   : 90
Number of days of warning before password expires : 7
```

▶ 8. 모든 사용자에 대해 현재 날짜부터 180일 후에 만료되도록 암호를 설정합니다. 관리 권한을 사용하여 구성 파일을 편집합니다.

- 8.1. **PASS_MAX_DAYS**에서 180를 **/etc/login.defs**로 설정합니다. 텍스트 편집기에서 파일을 열 때 관리 권한을 사용합니다. **vim /etc/login.defs** 명령을 사용하여 이 단계를 수행할 수 있습니다.

```

...output omitted...
# Password aging controls:
#
#      PASS_MAX_DAYS   Maximum number of days a password may be
#      used.
#      PASS_MIN_DAYS   Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN    Minimum acceptable password length.
#      PASS_WARN_AGE   Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS 180
PASS_MIN_DAYS  0
PASS_WARN_AGE  7
...output omitted...

```



중요

기본 암호 및 계정 만료 설정은 새 사용자에게 적용되고 기존 사용자에게는 적용되지 않습니다.

8.2. workstation 시스템에 student 사용자로 돌아갑니다.

```

[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$

```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-password
```

이것으로 섹션을 완료합니다.

▶ 랩

로컬 사용자 및 그룹 관리

이 랩에서는 기본 로컬 암호 정책을 설정하고, 사용자 3명으로 이루어진 보조 그룹을 생성하고, 해당 그룹이 **sudo**를 사용하여 **root**로 명령을 실행할 수 있도록 허용하고, 한 사용자의 암호 정책을 수정합니다.

결과

- 로컬 사용자 암호의 기본 암호 사용 기간 정책을 설정합니다.
- 보조 그룹을 생성하여 새 사용자에 대해 사용합니다.
- 새 보조 그룹을 사용하여 새 사용자 3명을 생성합니다.
- 생성된 사용자의 초기 암호를 설정합니다.
- **sudo** 명령을 사용하여 임의 사용자로 임의 명령을 실행하도록 보조 그룹 멤버를 구성합니다.
- 사용자 특정 암호 사용 기간 정책을 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start users-review
```

지침

1. **workstation** 시스템에서 **student** 사용자로 **serverb** 시스템에 대한 SSH 세션을 열고 **root** 사용자로 전환합니다.
2. **serverb** 시스템에서 새로 생성된 사용자가 **30일마다 암호를 변경해야 하도록 합니다.**
3. **GID가 35000인 consultants 그룹을 생성합니다.**
4. **모든 consultants 그룹 멤버가 임의 사용자로 임의 명령을 실행할 수 있도록 관리 권한을 구성합니다.** **/etc/sudoers** 파일을 편집하는데 **visudo**를 사용하지 마십시오. 대신 **/etc/sudoers.d** 디렉터리에 구성 파일을 배치합니다.
5. **consultants** 그룹을 보조 그룹으로 사용하여 **consultant1, consultant2, consultant3** 사용자를 생성합니다.
6. **consultant1, consultant2, consultant3** 암호를 **redhat**으로 설정합니다.
7. **consultant1, consultant2, consultant3** 계정이 현재 날짜부터 90일 후에 만료되도록 설정합니다.
8. **15일마다 암호를 변경하도록 consultant2 계정에 대한 암호 정책을 변경합니다.**
9. **또한 consultant1, consultant2, consultant3 사용자가 처음 로그인할 때 강제로 암호를 변경하도록 설정합니다.**

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade users-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

로컬 사용자 및 그룹 관리

이 랩에서는 기본 로컬 암호 정책을 설정하고, 사용자 3명으로 이루어진 보조 그룹을 생성하고, 해당 그룹이 **sudo**를 사용하여 **root**로 명령을 실행할 수 있도록 허용하고, 한 사용자의 암호 정책을 수정합니다.

결과

- 로컬 사용자 암호의 기본 암호 사용 기간 정책을 설정합니다.
- 보조 그룹을 생성하여 새 사용자에 대해 사용합니다.
- 새 보조 그룹을 사용하여 새 사용자 3명을 생성합니다.
- 생성된 사용자의 초기 암호를 설정합니다.
- **sudo** 명령을 사용하여 임의의 사용자로 임의의 명령을 실행하도록 보조 그룹 멤버를 구성합니다.
- 사용자 특정 암호 사용 기간 정책을 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start users-review
```

지침

1. **workstation** 시스템에서 **student** 사용자로 **serverb** 시스템에 대한 SSH 세션을 열고 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

2. **serverb** 시스템에서 새로 생성된 사용자가 30일마다 암호를 변경해야 하도록 합니다.

- 2.1. **/etc/login.defs** 파일에서 **PASS_MAX_DAYS**를 30으로 설정합니다. 텍스트 편집기에서 파일을 열 때 관리 권한을 사용합니다. **vim /etc/login.defs** 명령을 사용하여 이 단계를 수행할 수 있습니다.

```
...output omitted...
# Password aging controls:
#
#      PASS_MAX_DAYS    Maximum number of days a password may be
```

3장 | 로컬 사용자 및 그룹 관리

```
#      used.
#      PASS_MIN_DAYS   Minimum number of days allowed between
#      password changes.
#      PASS_MIN_LEN    Minimum acceptable password length.
#      PASS_WARN_AGE   Number of days warning given before a
#      password expires.
#
PASS_MAX_DAYS 30
PASS_MIN_DAYS  0
PASS_WARN_AGE   7
...output omitted...
```

3. GID가 35000인 **consultants** 그룹을 생성합니다.

```
[root@serverb ~]# groupadd -g 35000 consultants
```

4. 모든 **consultants** 그룹 멤버가 임의 사용자로 임의 명령을 실행할 수 있도록 관리 권한을 구성합니다. **/etc/sudoers** 파일을 편집하는데 **visudo**를 사용하지 마십시오. 대신 **/etc/sudoers.d** 디렉터리에 구성 파일을 배치합니다.

- 4.1. **/etc/sudoers.d/consultants** 파일을 생성하고 다음 콘텐츠를 파일에 추가합니다.
vim /etc/sudoers.d/consultants 명령을 사용하여 이 단계를 수행할 수 있습니다.

```
%consultants  ALL=(ALL) ALL
```

5. **consultants** 그룹을 보조 그룹으로 사용하여 **consultant1**, **consultant2**, **consultant3** 사용자를 생성합니다.

```
[root@serverb ~]# useradd -G consultants consultant1
[root@serverb ~]# useradd -G consultants consultant2
[root@serverb ~]# useradd -G consultants consultant3
```

6. **consultant1**, **consultant2**, **consultant3** 암호를 **redhat**으로 설정합니다.

```
[root@serverb ~]# passwd consultant1
Changing password for user consultant1.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
[root@serverb ~]# passwd consultant2
Changing password for user consultant2.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully
[root@serverb ~]# passwd consultant3
Changing password for user consultant3.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully
```

7. **consultant1, consultant2, consultant3** 계정이 현재 날짜부터 90일 후에 만료되도록 설정 합니다.

7.1. 90일 후의 날짜를 확인합니다. 이 반환된 날짜는 하나의 예입니다. 표시되는 값은 다음 단계에서 사용되며, 시스템의 현재 날짜 및 시간을 기반으로 합니다.

```
[root@serverb ~]# date -d "+90 days" +%F
2022-06-08
```

7.2. **consultant1, consultant2, consultant3** 계정의 계정 만료일을 앞의 단계에서 확인한 값과 동일한 값으로 설정합니다. 예를 들면 다음과 같습니다.

```
[root@serverb ~]# chage -E 2022-06-08 consultant1
[root@serverb ~]# chage -E 2022-06-08 consultant2
[root@serverb ~]# chage -E 2022-06-08 consultant3
```

8. 15일마다 암호를 변경하도록 **consultant2** 계정에 대한 암호 정책을 변경합니다.

```
[root@serverb ~]# chage -M 15 consultant2
```

9. 또한 **consultant1, consultant2, consultant3** 사용자가 처음 로그인할 때 강제로 암호를 변경하도록 설정합니다.

9.1. 사용자가 시스템에 처음 로그인할 때 암호를 변경해야 하도록 암호가 마지막으로 변경된 날을 0으로 설정합니다.

```
[root@serverb ~]# chage -d 0 consultant1
[root@serverb ~]# chage -d 0 consultant2
[root@serverb ~]# chage -d 0 consultant3
```

9.2. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade users-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish users-review
```

이것으로 섹션을 완료합니다.

요약

- Linux의 사용자 계정 유형은 수퍼유저, 시스템 사용자, 일반 사용자입니다.
- 사용자는 기본 그룹이 있으며 보조 그룹의 멤버일 수 있습니다.
- 중요한 `/etc/passwd`, `/etc/group`, `/etc/shadow` 파일에는 사용자 및 그룹 정보가 포함되어 있습니다.
- `su` 및 `sudo` 명령을 사용하여 수퍼유저로 명령을 실행할 수 있습니다.
- `useradd`, `usermod`, `userdel` 명령은 사용자를 관리합니다.
- `groupadd`, `groupmod`, `groupdel` 명령은 그룹을 관리합니다.
- `passwd` 명령은 사용자의 암호를 관리합니다.
- `chage` 명령은 사용자의 암호 만료 설정을 표시하고 구성합니다.

4장

파일에 대한 액세스 제어

목적

파일에 대해 Linux 파일 시스템 권한을 설정하고, 다양한 권한 설정에 따른 보안 영향을 해석합니다.

목표

- 명령줄 툴을 사용하여 파일의 권한 및 소유권을 변경 합니다.
- 사용자가 생성한 파일의 기본 권한을 제어하고, 특수 권한의 영향을 설명하고, 특수 권한 및 기본 권한을 사용하여 디렉터리에 생성된 파일의 그룹 소유자를 설정합니다.

섹션

- 명령줄에서 파일 시스템 권한 관리(안내에 따른 연습)
- 기본 권한 및 파일 액세스 관리(안내에 따른 연습)

랩

- 파일에 대한 액세스 제어

명령줄에서 파일 시스템 권한 관리

목표

명령줄 툴을 사용하여 파일의 권한 및 소유권을 변경합니다.

파일 및 디렉터리 권한 변경

chmod 명령은 다음 특성이 있습니다. 명령줄에서 파일 및 디렉터리 권한을 변경합니다. 이는 "모드 변경"으로 해석할 수 있는데 그 이유는 파일의 모드는 파일 권한의 또 다른 이름이기 때문입니다. 이는 사용 권한 지침과 변경할 파일 또는 디렉터리 목록을 차례로 표시합니다. 권한 명령은 심볼릭 방법 또는 8진수(숫자) 표기법으로 설정할 수 있습니다.

심볼릭 방법을 사용하여 권한 변경

chmod 명령을 사용하여 파일 및 디렉터리 권한을 수정합니다.

```
chmod Who/What/Which file|directory
```

다음 표에서 볼 수 있듯이, Who 는 사용자 클래스입니다. 사용자 클래스를 제공하지 않으면 **chmod** 명령은 all 그룹을 기본값으로 사용합니다.

Who	세트	설명
u	user	파일 소유자입니다.
g	group	파일 그룹의 멤버입니다.
o	other	파일 소유자나 파일 그룹의 멤버가 아닌 사용자입니다.
a	all	위 3개 그룹 모두

아래 표에서 볼 수 있듯이, What 은 Which를 수정하는 연산자입니다.

What	연산	설명
+	add	파일에 대한 권한을 추가합니다.
-	remove	파일에 대한 권한을 제거합니다.
=	set exactly	제공된 파일 권한을 정확하게 설정합니다.

Which 는 모드이며, 아래 표에서 볼 수 있듯이 파일 또는 디렉터리에 대한 권한을 지정합니다.

Which	모드	설명
r	read	파일에 대한 읽기 권한입니다. 디렉터리에 대한 권한을 표시합니다.
w	write	파일 또는 디렉터리에 대한 쓰기 권한입니다.

Which	모드	설명
x	execute	파일에 대한 실행 권한입니다. 디렉터리에 들어가서 디렉터리 내의 파일 및 하위 디렉터리에 액세스 할 수 있습니다.
x	special execute	디렉터리에 대한 실행 권한 또는 실행 비트가 하나 이상 설정된 경우 파일에 대한 실행 권한입니다.

파일 권한 변경을 위한 심볼릭 방법은 문자를 사용하여 권한 그룹을 표현합니다. **u**는 사용자, **g**는 그룹, **o**는 기타, **a**는 모두를 의미합니다.

심볼릭 방법을 사용하는 경우 새 권한 그룹 전체를 설정할 필요는 없습니다. 대신, 기존 권한 중 하나 이상을 변경할 수 있습니다. 더하기(+) 또는 빼기(-) 문자를 사용하여 각각 권한을 추가 또는 제거하거나, 등호(=) 문자를 사용하여 권한 그룹의 전체 세트를 교체합니다.

단일 문자는 권한 자체를 나타냅니다. **r**은 읽기, **w**는 쓰기, **x**는 실행을 나타냅니다. 파일이 디렉터리이거나 사용자, 그룹 또는 기타에 대해 실행이 이미 설정된 경우에만 대문자 **X**를 권한 플래그로 사용하여 실행 권한을 추가할 수 있습니다.

다음 목록은 심볼릭 방법을 사용하여 권한을 변경하는 몇 가지 예제를 보여줍니다.

document.pdf 파일에 대한 그룹 및 기타의 읽기 및 쓰기 권한을 제거합니다.

```
[user@host ~]$ chmod go-rw document.pdf
```

myscript.sh 파일에 대한 모든 사용자의 실행 권한을 추가합니다.

```
[user@host ~]$ chmod a+x myscript.sh
```

chmod 명령의 **-R** 옵션을 사용하여 전체 디렉터리 트리의 파일에 대한 권한을 재귀적으로 설정할 수 있습니다. 예를 들어 다음 명령은 **myfolder** 디렉터리와 이 디렉터리에 포함된 파일 및 디렉터리를 소유하는 그룹의 멤버에 대한 읽기, 쓰기, 실행 권한을 재귀적으로 추가합니다.

```
[user@host ~]$ chmod -R g+rwx /home/user/myfolder
```

chmod 명령의 **-R** 옵션을 **-X** 옵션과 함께 사용하여 심볼릭 방법으로 권한을 설정할 수도 있습니다. **chmod** 명령의 **X** 옵션을 사용하면 대부분의 파일에 대한 권한을 변경하지 않고도 파일 내용에 액세스할 수 있도록 디렉터리에 대한 실행(검색) 권한을 설정할 수 있습니다. 그러나 **X** 옵션을 사용할 때는 주의하십시오. 파일에 실행 권한이 설정되어 있는 경우 **X** 옵션을 사용하면 해당 파일에도 지정한 실행 권한이 설정되기 때문입니다.

예를 들어 다음 명령은 그룹 소유자의 경우 **demodir** 디렉터리 및 모든 하위 항목에 대한 읽기 및 쓰기 액세스 권한을 재귀적으로 설정하지만 사용자, 그룹 또는 기타의 경우 실행 권한이 이미 설정된 디렉터리와 파일에만 그룹 실행 권한을 적용합니다.

```
[root@host opt]# chmod -R g+rwx demodir
```

8진수 방법을 사용하여 권한 변경

chmod 명령을 사용하여 심볼릭 방법 대신 8진수 방법으로 파일 권한을 변경할 수 있습니다. 다음 예제에서 # 문자는 숫자를 나타냅니다.

```
chmod ### file|directory
```

8진수 방법을 사용할 경우 권한을 세 자리(고급 권한을 설정할 때는 네 자리) 8진수로 나타낼 수 있습니다. 한 자리 8진수는 0-7의 단일 값을 나타낼 수 있습니다.

권한을 세 자리 8진수로 나타낼 때 각 숫자는 하나의 액세스 수준에 해당하며, 왼쪽부터 오른쪽으로 사용자, 그룹, 기타입니다. 각 자릿수를 결정하려면

- 0으로 시작하십시오.
- 이 액세스 수준에 읽기 권한을 추가하려면 4를 더합니다.
- 쓰기 권한을 추가하려면 2를 더합니다.
- 실행 권한을 추가하려면 1을 더합니다.

다음 다이어그램은 시스템에서 **644** 8진수 권한 값을 해석하는 방식을 보여줍니다.

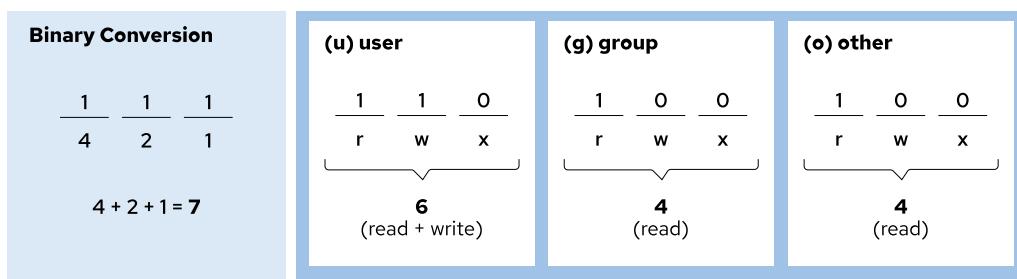


그림 4.1: 8진수 방법의 시각적 표시

숙련된 관리자는 8진수 권한을 사용하는 경우가 많습니다. 단일 또는 일치하는 파일에서 구현할 수 있으면서도 전체 권한 제어가 가능하기 때문입니다.

다음 목록은 8진수 방법을 사용하여 권한을 변경하는 몇 가지 예제를 보여줍니다.

사용자의 경우 **sample.txt** 파일에 대한 읽기 및 쓰기 권한을 설정하고, 그룹 및 기타의 경우 읽기 권한을 설정합니다.

```
[user@host ~]$ chmod 644 sample.txt
```

사용자의 경우 **sampledir** 디렉터리에 대한 읽기, 쓰기, 실행 권한을 설정하고, 그룹의 경우 읽기 및 실행 권한을 설정하고, 기타의 경우 아무 권한도 설정하지 않습니다.

```
[user@host ~]$ chmod 750 sampledir
```

파일 및 디렉터리 사용자 또는 그룹 소유권 변경

사용자는 생성한 파일을 소유합니다. 기본적으로 새 파일의 그룹 소유권은 해당 파일을 생성한 사용자의 기본 그룹입니다. Red Hat Enterprise Linux에서 사용자의 기본 그룹은 일반적으로 멤버가 해당 사용자뿐인 개인 그룹입니다. 그룹 멤버십에 따라 파일에 대한 액세스 권한을 부여하기 위해 파일을 소유한 그룹을 변경해야 할 수도 있습니다.

root 사용자만 파일을 소유한 사용자를 변경할 수 있습니다. 그러나 파일의 소유자와 **root** 사용자가 그룹 소유권을 설정할 수 있습니다. **root** 사용자는 모든 그룹에 파일 소유권을 부여할 수 있지만, 일반 사용자만 대상 그룹의 멤버인 경우 파일의 그룹 소유권을 변경할 수 있습니다.

4장 | 파일에 대한 액세스 제어

chown(소유자 변경) 명령을 사용하여 파일 소유권을 변경할 수 있습니다. 예를 들어 **app.conf** 파일의 소유권을 **student** 사용자에게 부여하려면 다음 명령을 사용합니다.

```
[root@host ~]# chown student app.conf
```

chown 명령의 **-R** 옵션은 전체 디렉터리 트리의 소유권을 재귀적으로 변경합니다. 다음 명령은 **Pictures** 디렉터리와 디렉터리 내의 모든 파일 및 하위 디렉터리의 소유권을 **student** 사용자에게 부여합니다.

```
[root@host ~]# chown -R student Pictures
```

chown 명령을 사용하면 그룹 이름 앞에 콜론(:)을 추가하여 파일의 그룹 소유권을 변경할 수도 있습니다. 예를 들어 다음 명령은 **Pictures** 디렉터리의 그룹 소유권을 **admins**로 변경합니다.

```
[root@host ~]# chown :admins Pictures
```

chown 명령을 사용하면 owner:group 구문으로 소유자와 그룹을 동시에 변경할 수 있습니다. 예를 들어 **Pictures** 디렉터리의 소유권을 **visitor** 사용자로 변경하고 그룹을 **guests**로 변경하려면 다음 명령을 사용합니다.

```
[root@host ~]# chown visitor:guests Pictures
```

chown 명령을 사용하는 대신 일부 사용자는 **chgrp** 명령을 사용하여 그룹 소유권을 변경합니다. 이 명령은 **chown**과 유사하게 작동합니다. 단, 그룹 소유권을 변경하는 데만 사용할 수 있으며 그룹 이름 앞의 콜론(:)이 필요하지 않습니다.



중요

콜론 대신 마침표로 소유자와 그룹을 구분하는 대체 **chown** 구문을 발견할 수도 있습니다.

```
[root@host ~]# chown owner.group filename
```

이 구문을 사용하지 않고 항상 콜론을 사용하는 것이 좋습니다. 마침표는 사용자 이름에서 유 효한 문자이므로 **chown** 명령이 의도를 잘못 해석할 수 있습니다. 명령에서 사용자와 그룹을 파일 이름으로 해석할 수도 있습니다. 대신, 사용자와 그룹을 동시에 설정할 때는 콜론 문자만 사용합니다.



참조

ls(1), **chmod(1)**, **chown(1)** 및 **chgrp(1)** 도움말 페이지

▶ 연습 가이드

명령줄에서 파일 시스템 권한 관리

이 연습에서는 파일 시스템 권한을 사용하여 특정 그룹의 모든 멤버가 파일을 추가하고 삭제할 수 있는 디렉터리를 생성합니다.

결과

- 특정 그룹의 모든 멤버가 액세스할 수 있는 협업 디렉터리를 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start perms-cli
```

지침

- ▶ 1. **workstation**에서 **student** 사용자로 **servera**에 로그인한 다음, **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **/home/consultants** 디렉터리를 생성합니다.

```
[root@servera ~]# mkdir /home/consultants
```

- ▶ 3. **consultants** 디렉터리의 그룹 소유권을 **consultants**로 변경합니다.

```
[root@servera ~]# chown :consultants /home/consultants
```

- ▶ 4. 그룹 멤버가 **/home/consultants** 디렉터리에서 파일을 생성하고 삭제할 수 있도록 **consultants** 그룹의 권한을 수정합니다.

현재 권한에서는 다른 사용자가 파일에 액세스할 수 없습니다. 적절한 권한을 설정해야 합니다.

4.1. **consultants** 그룹의 권한이 그룹 멤버가 **/home/consultants** 디렉터리에서 파일을 생성하고 삭제할 수 있도록 허용하는지 확인합니다.

consultants 그룹에는 현재 쓰기 권한이 없습니다.

```
[root@servera ~]# ls -ld /home/consultants
drwxr-xr-x. 2 root     consultants   6 Mar  1 12:08 /home/consultants
```

- 4.2. **consultants** 그룹에 쓰기 권한을 추가합니다. 심볼릭 방법을 사용하여 적절한 권한을 설정합니다.

```
[root@servera ~]# chmod g+w /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwxr-x. 2 root consultants 6 Mar  1 13:21 /home/consultants
```

- 4.3. 기타 사용자가 **/home/consultants** 디렉터리의 파일에 액세스할 수 없도록 금지합니다. 8진수 방법을 사용하여 적절한 권한을 설정합니다.

```
[root@servera ~]# chmod 770 /home/consultants
[root@servera ~]# ls -ld /home/consultants
drwxrwx---. 2 root consultants 6 Mar  1 12:08 /home/consultants/
```

- ▶ 5. **root** 쉘을 종료하고 **consultant1** 사용자로 전환합니다. 암호는 **redhat**입니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ su - consultant1
Password: redhat
[consultant1@servera ~]$
```

- ▶ 6. **/home/consultants** 디렉터리로 이동하여 **consultant1.txt**라는 파일을 생성합니다.

- 6.1. **/home/consultants** 디렉터리로 변경합니다.

```
[consultant1@servera ~]$ cd /home/consultants
```

- 6.2. **consultant1.txt**라는 빈 폴더를 생성합니다.

```
[consultant1@servera consultants]$ touch consultant1.txt
```

- ▶ 7. 새 파일의 기본 사용자 및 그룹 소유권과 해당 권한을 표시합니다.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultant1 0 Mar  1 12:53 consultant1.txt
```

- ▶ 8. **consultants** 그룹의 모든 멤버가 **consultant1.txt** 파일을 편집할 수 있게 합니다. **consultant1.txt** 파일의 그룹 소유권을 **consultants**로 변경합니다.

- 8.1. **chown** 명령을 사용하여 **consultant1.txt** 파일의 그룹 소유권을 **consultants**로 변경합니다.

```
[consultant1@servera consultants]$ chown :consultants consultant1.txt
```

8.2. **consultant1.txt** 파일의 새 소유권을 표시합니다.

```
[consultant1@servera consultants]$ ls -l consultant1.txt
-rw-rw-r--. 1 consultant1 consultants 0 Mar 1 12:53 consultant1.txt
```

▶ 9. 헬을 종료하고 **consultant2** 사용자로 전환합니다. 암호는 **redhat**입니다.

```
[consultant1@servera consultants]$ exit
logout
[student@servera ~]$ su - consultant2
Password: redhat
[consultant2@servera ~]$
```

▶ 10. **/home/consultants** 디렉터리로 이동합니다. **consultant2** 사용자가 **consultant1.txt** 파일에 내용을 추가할 수 있게 합니다.

10.1. **/home/consultants** 디렉터리로 변경합니다. **consultant1.txt** 파일에 **text**를 추가합니다.

```
[consultant2@servera ~]$ cd /home/consultants/
[consultant2@servera consultants]$ echo "text" >> consultant1.txt
```

10.2. **consultant1.txt** 파일에 텍스트가 있는지 확인합니다.

```
[consultant2@servera consultants]$ cat consultant1.txt
text
```

10.3. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[consultant2@servera consultants]$ exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish perms-cli
```

이것으로 섹션을 완료합니다.

기본 권한 및 파일 액세스 관리

목표

사용자가 생성한 파일의 기본 권한을 제어하고, 특수 권한의 영향을 설명하고, 특수 권한 및 기본 권한을 사용하여 디렉터리에 생성된 파일의 그룹 소유자를 설정합니다.

특수 권한

특수 권한은 사용자, 그룹 및 기타 유형에 더하여 네 번째 권한 유형입니다. 이름에서 알 수 있듯이, 특수 권한은 기본 권한 유형에서 허용하는 기능 이외의 추가 액세스 관련 기능을 제공합니다. 이 섹션에서는 아래 표에 요약된 특수 권한의 영향에 대해 설명합니다.

특수 권한이 파일 및 디렉터리에 미치는 영향

권한	파일에 미치는 영향	디렉터리에 미치는 영향
u+s(suid)	파일을 실행한 사용자가 아니라 파일을 소유한 사용자로 파일이 실행됩니다.	아무 영향이 없습니다.
g+s(sgid)	파일은 파일을 소유한 그룹 권한으로 실행됩니다.	디렉터리에 생성된 파일의 그룹 소유자가 디렉터리의 그룹 소유자와 일치합니다.
o+t (sticky)	아무 영향이 없습니다.	디렉터리에 대한 쓰기 권한이 있는 사용자가 자신이 소유한 파일만 제거할 수 있고 다른 사용자가 소유한 파일을 제거하거나 강제로 저장할 수 없습니다.

실행 파일에 대한 setuid 권한은 명령을 실행한 사용자가 아니라 해당 파일을 소유한 사용자로 명령이 실행됨을 의미합니다. 한 가지 예는 **passwd** 명령입니다.

```
[user@host ~]$ ls -l /usr/bin/passwd
-rwsr-xr-x. 1 root root 35504 Jul 16 2010 /usr/bin/passwd
```

긴 목록에서 일반적으로 x 문자(소유자 실행 권한)가 올 것으로 예상하는 위치에 소문자 s가 있으면 **setuid** 권한임을 확인할 수 있습니다. 소유자에게 실행 권한이 없으면 이 문자가 대문자 S로 교체됩니다.

디렉터리에 대한 setgid 특수 권한은 디렉터리에 생성된 파일이 생성하는 사용자로부터 그룹 소유권을 상속하지 않고 디렉터리에서 그룹 소유권을 상속함을 의미합니다. 이 기능은 일반적으로 특정 그룹이 항상 디렉터리의 파일을 소유해야 하는 경우 또는 파일을 기본 개인 그룹에서 공유 그룹으로 자동 변경하기 위해 그룹 협업 디렉터리에서 사용됩니다. 이 동작의 예로 **/run/log/journal** 디렉터리가 있습니다.

```
[user@host ~]$ ls -ld /run/log/journal
drwxr-sr-x. 3 root systemd-journal 60 May 18 09:15 /run/log/journal
```

실행 파일에 **setgid**가 설정된 경우 명령을 실행한 사용자가 아니라 해당 파일을 소유한 그룹으로 명령이 실행됩니다. 이 상태는 **setuid**의 작동 방식과 비슷합니다. 한 가지 예는 **locate** 명령입니다.

4장 | 파일에 대한 액세스 제어

```
[user@host ~]$ ls -ld /usr/bin/locate
-rwx--s--x. 1 root slocate 47128 Aug 12 17:17 /usr/bin/locate
```

긴 목록에서 일반적으로 **x** 문자(그룹 실행 권한)가 올 것으로 예상하는 위치에 소문자 **s**가 있으면 **setgid** 권한임을 확인할 수 있습니다. 그룹에 실행 권한이 없으면 이 문자가 대문자 **S**로 교체됩니다.

최종적으로, 디렉터리의 스티키 비트는 파일 삭제에 대한 특수 제한을 설정합니다. 파일 소유자(및 **root** 사용자)만 디렉터리 내의 파일을 삭제할 수 있습니다. 예를 들어 **/tmp** 디렉터리가 있습니다.

```
[user@host ~]$ ls -ld /tmp
drwxrwxrwt. 39 root root 4096 Feb 8 20:52 /tmp
```

긴 목록에서 일반적으로 **x** 문자(기타 실행 권한)가 올 것으로 예상하는 위치에 소문자 **t**가 있으면 스티키 권한임을 확인할 수 있습니다. 기타에 실행 권한이 없으면 이 문자가 대문자 **T**로 교체됩니다.

특수 권한 설정

- **심볼릭**: **setuid = u+s; setgid = g+s; sticky = o+t**
- **8진수**: 앞에서 더한 네 번째 숫자. **setuid = 4; setgid = 2; sticky = 1**

특수 권한의 예

심볼릭 방법을 사용하여 **example** 디렉터리에서 **setgid** 비트를 추가합니다.

```
[user@host ~]# chmod g+s example
```

심볼릭 방법을 사용하여 **example** 디렉터리에서 **setuid** 비트를 제거합니다.

```
[user@host ~]# chmod u-s example
```

8진수 방법을 사용하여 **example** 디렉터리에 **setgid** 비트를 설정하고 사용자 및 그룹의 경우 읽기, 쓰기, 실행 권한을 추가합니다. 기타의 경우 액세스 권한을 추가하지 않습니다.

```
[user@host ~]# chmod 2770 example
```

8진수 방법을 사용하여 **example** 디렉터리에서 **setgid** 비트를 제거하고 사용자 및 그룹의 경우 읽기, 쓰기, 실행 권한을 추가합니다. 기타의 경우 액세스 권한을 추가하지 않습니다. 8진수 방법을 사용하여 특수 권한을 제거할 때는 권한 값의 시작 부분에 **0**을 추가해야 합니다.

```
[user@host ~]# chmod 0770 example
```

기본 파일 권한

생성 시 파일에 초기 권한이 할당됩니다. 초기 권한에 영향을 주는 것은 두 가지 요소입니다. 첫 번째는 정규 파일 또는 디렉터리 생성 여부입니다. 두 번째는 현재 umask로, 사용자 파일 생성 마스크를 나타냅니다.

디렉터리를 생성하는 경우 초기 8진수 권한은 0777(**drwxrwxrwx**)입니다. 일반 파일을 생성하는 경우 초기 8진수 권한은 0666(-**rwx-rwx-rw-**)입니다. 일반 파일에는 항상 실행 권한을 명시적으로 추가해야 합니다. 이 단계를 수행하면 공격자가 시스템을 손상시키고 악성 파일을 생성하여 실행하기가 더 어려워집니다.

4장 | 파일에 대한 액세스 제어

또한 쉘 세션은 umask를 설정하여 파일의 초기 권한을 더욱 제한합니다. umask는 프로세스에서 생성하는 새 파일과 디렉터리의 권한을 지우는 8진수 비트 마스크입니다. umask에 비트가 설정된 경우 해당 권한이 새 파일에서 제거됩니다. 예를 들어 umask 0002는 기타 사용자의 쓰기 비트를 지웁니다. 선행 0은 특수, 사용자 및 그룹 권한이 지워지지 않았음을 나타냅니다. umask가 0077인 경우 새로 생성된 파일의 모든 그룹 및 기타 권한이 제거됩니다.

인수가 없는 **umask** 명령은 현재 쉘의 umask 값을 표시합니다.

```
[user@host ~]$ umask  
0022
```

umask 명령에 단일 8진수 인수를 사용하여 현재 쉘의 umask를 변경합니다. 인수는 새 umask 값에 해당하는 8진수 값이어야 합니다. umask에서 선행 0을 생략할 수 있습니다. 예를 들어 **umask 077**은 **umask 0077**과 동일합니다.

Bash 쉘 사용자의 시스템 기본 umask 값은 **/etc/login.defs** 파일과 **/etc/bashrc** 파일에 정의됩니다. 사용자는 홈 디렉터리의 **.bash_profile** 또는 **.bashrc** 파일에서 시스템 기본값을 재정의할 수 있습니다.



중요

Red Hat Enterprise Linux 8과 그 이전 버전에서 사용자 계정의 UID가 200 이상이고 사용자 이름과 해당 계정의 기본 그룹 이름이 같으면 기본 umask는 0002입니다. 그렇지 않으면 기본 umask는 0022입니다.

Red Hat Enterprise Linux 9에서는 모든 계정의 umask가 0022가 되도록 기본 unmask를 변경하는 중입니다. RHEL 9.0 및 9.1에서 로그인 쉘을 시작하면 umask는 0022입니다. 그러나 대화형 비로그인 쉘을 시작할 때(예: 그래픽 UI에서 **gnome-terminal**을 시작할 때) 계정의 UID가 200 이상이고 기본 그룹의 이름이 사용자 계정과 동일한 경우 umask는 0002입니다. 향후 Red Hat Enterprise Linux 9 버전에서 모든 상황의 umask 기본값이 0022가 되도록 기본 unmask가 변경될 예정입니다.

Bugzilla 문제 https://bugzilla.redhat.com/show_bug.cgi?id=2062601 에서는 RHEL 9 동작에서 이 변경 사항을 추적합니다.

umask 유ти리티가 권한에 미치는 영향

다음 예제에서는 umask가 파일 및 디렉터리의 권한에 미치는 영향을 설명합니다. 현재 쉘에서 파일과 디렉터리 둘 다에 대한 기본 umask 권한을 확인합니다.



중요

다음 예제에서는 쉘의 umask가 0022로 설정되었다고 가정합니다.

일반 파일을 생성하는 경우 초기 8진수 권한은 0666(바이너리 표시의 000 110 110 110)입니다. 그런 다음 0022 umask(000 000 010 010)가 그룹 및 기타의 쓰기 권한 비트를 비활성화합니다. 따라서 소유자는 파일에 대한 읽기 및 쓰기 권한이 모두 있고, 그룹 및 기타는 둘 다 읽기(000 110 100 100)로 설정됩니다.

	Symbolic	Numeric octal	Numeric binary
Initial file permissions	rw-rw-rw-	0666	000 110 110 110
umask	-----w-	0002	000 000 000 010
Resulting file permissions	rw-rw-r--	0664	000 110 110 100

그림 4.2: 파일에 대한 umask 계산의 예

```
[user@host ~]$ umask
0022
[user@host ~]$ touch default.txt
[user@host ~]$ ls -l default.txt
-rw-r--r--. 1 user user 0 May  9 01:54 default.txt
```

디렉터리를 생성하는 경우 초기 8진수 권한은 0777(000 111 111 111)입니다. 그런 다음 0022 umask(000 000 010 010)가 그룹 및 기타의 쓰기 권한 비트를 비활성화합니다. 따라서 소유자는 디렉터리에 대한 읽기, 쓰기, 실행 권한이 있고, 그룹 및 기타는 둘 다 읽기 및 실행(000 111 101 101)으로 설정됩니다.

	Symbolic	Numeric octal	Numeric binary
Initial directory permissions	rwxrwxrwx	0777	000 111 111 111
umask	-----w-	0002	000 000 000 010
Resulting directory permissions	rwxrwxr-x	0775	000 111 111 101

그림 4.3: 디렉터리에 대한 umask 계산의 예

```
[user@host ~]$ umask
0022
[user@host ~]$ mkdir default
[user@host ~]$ ls -ld default
drwxr-xr-x. 2 user user 0 May  9 01:54 default
```

umask 값을 0으로 설정하면 기타의 파일 권한이 읽기에서 읽기 및 쓰기로 변경됩니다. 기타의 디렉터리 권한은 읽기 및 실행에서 읽기, 쓰기, 실행으로 변경됩니다.

```
[user@host ~]$ umask 0
[user@host ~]$ touch zero.txt
[user@host ~]$ ls -l zero.txt
-rw-rw-rw-. 1 user user 0 May  9 01:54 zero.txt
[user@host ~]$ mkdir zero
[user@host ~]$ ls -ld zero
drwxrwxrwx. 2 user user 0 May  9 01:54 zero
```

기타의 파일 및 디렉터리 권한을 모두 마스크하려면 umask 값을 007로 설정합니다.

```
[user@host ~]$ umask 007
[user@host ~]$ touch seven.txt
[user@host ~]$ ls -l seven.txt
-rw-rw----. 1 user user 0 May  9 01:55 seven.txt
[user@host ~]$ mkdir seven
[user@host ~]$ ls -ld seven
drwxrwx---. 2 user user 0 May  9 01:54 seven
```

umask가 027이면 사용자는 새 파일에 대한 읽기 및 쓰기 권한을 갖고, 그룹은 읽기 권한을 갖습니다. 그룹은 새 디렉터리에 대한 읽기 및 실행 권한을 갖고, 기타는 권한이 없습니다.

```
[user@host ~]$ umask 027
[user@host ~]$ touch two-seven.txt
[user@host ~]$ ls -l two-seven.txt
-rw-r-----. 1 user user 0 May  9 01:55 two-seven.txt
[user@host ~]$ mkdir two-seven
[user@host ~]$ ls -ld two-seven
drwxr-x---. 2 user user 0 May  9 01:54 two-seven
```

기본 권한 변경

Red Hat Enterprise Linux 9에서 `/etc/login.defs` 파일은 사용자의 기본 umask를 설정합니다. 기본적으로 **UMASK** 행은 기본 umask를 0022로 지정합니다.

Red Hat Enterprise Linux 9.0 및 9.1에서는 이 파일이 로그인 쉘에만 영향을 미칩니다. 새 터미널 창을 시작하거나 다른 방식으로 대화형 비로그인 쉘을 시작하는 경우 `/etc/bashrc`의 설정이 계속 적용됩니다. 이러한 쉘의 경우 계정의 UID가 200 이상이고 사용자 이름과 기본 그룹 이름이 같으면 계정에 umask 0002가 할당됩니다. 그렇지 않으면 umask는 0022입니다.

`root` 사용자는 `/etc/profile.d/` 디렉터리에 `local-umask.sh` 쉘 시작 스크립트를 추가하여 대화형 비로그인 쉘의 기본 umask를 변경할 수 있습니다. 다음 예제에서는 `local-umask.sh` 파일을 보여줍니다.

```
[root@host ~]# cat /etc/profile.d/local-umask.sh
# Overrides default umask configuration asda sda
if [ $UID -gt 199 ] && [ "`id -gn`" = "`id -un`" ]; then
    umask 007
else
    umask 022
fi
```

앞의 예제는 UID가 199보다 크고 사용자 이름과 기본 그룹 이름이 일치하는 사용자의 경우 umask를 0007로 설정하고 다른 모든 사용자의 경우 0022로 설정합니다. (선행 0은 생략할 수 있습니다.) 모든 사용자의 umask를 0022로 설정하려면 다음 콘텐츠로 해당 파일을 생성합니다.

```
# Overrides default umask configuration
umask 022
```

쉘에서 로그아웃한 후 다시 로그인하거나 `umask` 명령을 사용하여 수동으로 변경할 때까지 쉘의 현재 umask가 적용됩니다.



참조

`bash(1)`, `ls(1)`, `chmod(1)` 및 `umask(1)` 도움말 페이지

▶ 연습 가이드

기본 권한 및 파일 액세스 관리

이 연습에서는 umask 설정과 **setgid** 권한을 사용하여 디렉터리에 생성된 파일에 대한 권한을 제어합니다.

결과

- **operators** 그룹이 새 파일을 자동으로 소유하게 되는 공유 디렉터리를 생성합니다.
- 다양한 umask 설정으로 실험합니다.
- 특정 사용자에 대한 기본 권한을 조정합니다.
- 변경 사항을 확인합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start perms-default
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 암호로 **redhat**을 사용하여 **operator1** 사용자로 전환합니다.

```
[student@servera ~]$ su - operator1
Password: redhat
[operator1@servera ~]$
```

- ▶ 3. **operator1** 사용자의 기본 umask 값을 표시합니다.

```
[operator1@servera ~]$ umask
0022
```

- ▶ 4. **/tmp/shared** 디렉터리를 생성합니다. **/tmp/shared** 디렉터리에 **defaults** 파일을 생성합니다. 기본 권한을 확인합니다.

- 4.1. **/tmp/shared** 디렉터리를 생성합니다. 새 디렉터리의 권한을 표시합니다.

4장 | 파일에 대한 액세스 제어

```
[operator1@servera ~]$ mkdir /tmp/shared
[operator1@servera ~]$ ls -ld /tmp/shared
drwxr-xr-x. 2 operator1 operator1 6 Feb 4 14:06 /tmp/shared
```

4.2. `/tmp/shared` 디렉터리에 `defaults` 파일을 생성합니다.

```
[operator1@servera ~]$ touch /tmp/shared/defaults
```

4.3. 새 파일의 권한을 표시합니다.

```
[operator1@servera ~]$ ls -l /tmp/shared/defaults
-rw-r--r--. 1 operator1 operator1 0 Feb 4 14:09 /tmp/shared/defaults
```

▶ 5. `/tmp/shared` 디렉터리의 그룹 소유권을 `operators` 그룹으로 변경합니다. 새 소유권과 권한을 확인합니다.

5.1. `/tmp/shared` 디렉터리의 그룹 소유권을 `operators` 그룹으로 변경합니다.

```
[operator1@servera ~]$ chown :operators /tmp/shared
```

5.2. `/tmp/shared` 디렉터리의 권한을 표시합니다.

```
[operator1@servera ~]$ ls -ld /tmp/shared
drwxr-xr-x. 2 operator1 operators 22 Feb 4 14:09 /tmp/shared
```

5.3. `/tmp/shared` 디렉터리에 `group` 파일을 생성합니다. 파일 권한을 표시합니다.

```
[operator1@servera ~]$ touch /tmp/shared/group
[operator1@servera ~]$ ls -l /tmp/shared/group
-rw-r--r--. 1 operator1 operator1 0 Feb 4 17:00 /tmp/shared/group
```



참고

`/tmp/shared/group` 파일의 그룹 소유자는 `operators`가 아니라 `operator1`입니다.

▶ 6. `operators` 그룹이 `/tmp/shared` 디렉터리에 생성된 파일을 소유하게 합니다.

6.1. `/tmp/shared` 디렉터리의 그룹 ID를 `operators` 그룹으로 설정합니다.

```
[operator1@servera ~]$ chmod g+s /tmp/shared
```

6.2. `/tmp/shared` 디렉터리에 `ops_db.txt` 파일을 생성합니다.

```
[operator1@servera ~]$ touch /tmp/shared/ops_db.txt
```

6.3. `operators` 그룹이 새 파일의 그룹 소유자인지 확인합니다.

```
[operator1@servera ~]$ ls -l /tmp/shared/ops_db.txt
-rw-r--r--. 1 operator1 operators 0 Feb  4 16:11 /tmp/shared/ops_db.txt
```

- ▶ 7. /tmp/shared 디렉터리에 ops_net.txt 파일을 생성합니다. 소유권과 권한을 기록합니다. operator1 사용자의 umask를 변경합니다. ops_prod.txt 파일을 생성합니다. ops_prod.txt 파일의 소유권과 권한을 기록합니다.

- 7.1. /tmp/shared 디렉터리에 ops_net.txt 파일을 생성합니다.

```
[operator1@servera ~]$ touch /tmp/shared/ops_net.txt
```

- 7.2. ops_net.txt 파일의 권한을 표시합니다.

```
[operator1@servera ~]$ ls -l /tmp/shared/ops_net.txt
-rw-r--r--. 1 operator1 operators 5 Feb  5 15:43 /tmp/shared/ops_net.txt
```

- 7.3. operator1 사용자의 umask를 027로 변경합니다. 변경 사항을 확인합니다.

```
[operator1@servera ~]$ umask 027
[operator1@servera ~]$ umask
0027
```

- 7.4. /tmp/shared/ 디렉터리에 ops_prod.txt 파일을 생성합니다. 새로 생성된 파일에서 operators 그룹의 경우 읽기 전용 권한이 있고, 기타 사용자의 경우 액세스 권한이 없는지 확인합니다.

```
[operator1@servera ~]$ touch /tmp/shared/ops_prod.txt
[operator1@servera ~]$ ls -l /tmp/shared/ops_prod.txt
-rw-r-----. 1 operator1 operators 0 Feb  5 15:56 /tmp/shared/ops_prod.txt
```

- ▶ 8. 새 터미널 창을 열고 servera에 operator1로 로그인합니다.

```
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$
```

- ▶ 9. operator1의 umask 값을 나열합니다.

```
[operator1@servera ~]$ umask
0022
```

- ▶ 10. operator1 사용자의 기본 umask를 변경합니다. 새 umask는 그룹에 속하지 않은 사용자의 모든 액세스를 금지합니다. umask가 변경되었는지 확인합니다.

- 10.1. operator1 사용자의 기본 umask를 007로 변경합니다.

```
[operator1@servera ~]$ echo "umask 007" >> ~/.bashrc
[operator1@servera ~]$ cat ~/.bashrc
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi
...output omitted...
umask 007
```

10.2. 로그아웃한 다음, **operator1** 사용자로 다시 로그인합니다. 변경 사항이 영구적인지 확인합니다.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh operator1@servera
...output omitted...
[operator1@servera ~]$ umask
0007
```

- ▶ 11. **/tmp/shared/** 디렉터리에 **ops_prod2.txt** 파일을 생성합니다. 새 umask 007로 인해 새로 생성된 파일에 대해 **operators** 그룹에 읽기 및 쓰기 액세스 권한이 있고, 기타 사용자에는 액세스 권한이 없는지 확인합니다.

```
[operator1@servera ~]$ touch /tmp/shared/ops_prod2.txt
[operator1@servera ~]$ ls -l /tmp/shared/ops_prod2.txt
-rw-rw----. 1 operator1 operators 0 Feb  0 15:56 /tmp/shared/ops_prod2.txt
```

- ▶ 12. **servera**에서 **operator1** 및 **student** 사용자 쉘을 모두 닫습니다. **workstation** 시스템에 **student** 사용자로 돌아갑니다.



경고

모든 **operator1** 쉘을 종료하지 않으면 종료 스크립트가 실패합니다.

```
[operator1@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish perms-default
```

이것으로 섹션을 완료합니다.

▶ 랩

파일에 대한 액세스 제어

이 랩에서는 파일에 대한 권한을 구성하고, 특정 그룹의 사용자가 로컬 파일 시스템의 파일을 공유하는데 사용할 수 있는 디렉터리를 설정합니다.

결과

- 사용자가 공동으로 파일 작업을 할 수 있는 디렉터리를 생성합니다.
- 그룹 소유권이 자동으로 할당되는 파일을 생성합니다.
- 그룹 외부에서 액세스할 수 없는 파일을 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start perms-review
```

지침

- student** 사용자로 **serverb**에 로그인합니다. 쉘 프롬프트에서 **sudo -i** 명령을 실행하여 **root** 사용자가 됩니다. **student**를 **student** 사용자 암호로 사용합니다.
- /home/techdocs** 디렉터리를 생성합니다.
- /home/techdocs** 디렉터리의 그룹 소유권을 **techdocs** 그룹으로 변경합니다.
- techdocs** 그룹의 사용자가 **/home/techdocs** 디렉터리에 파일을 생성할 수 없는지 확인합니다.
- /home/techdocs** 디렉터리에 대한 권한을 설정합니다. **/home/techdocs** 디렉터리에서 **setgid(2)**, 소유자/사용자 및 그룹의 경우 읽기, 쓰기, 실행 권한(7), 기타 사용자의 경우 권한 없음(0)을 구성합니다.
- 권한이 정확하게 설정되었는지 확인합니다.
이제 **techdocs** 그룹에 쓰기 권한이 있습니다.
- 이제 **techdocs** 그룹의 사용자가 **/home/techdocs** 디렉터리에 파일을 생성하고 편집할 수 있는지 확인합니다. **techdocs** 그룹에 속하지 않은 사용자는 **/home/techdocs** 디렉터리에 파일을 생성하거나 편집할 수 없습니다. **tech1** 및 **tech2** 사용자는 **techdocs** 그룹에 속해 있습니다. **database1** 사용자는 해당 그룹에 속하지 않습니다.
- /etc/login.defs** 파일을 수정하여 로그인 쉘의 기본 umask를 조정합니다. 일반 사용자에게는 해당 사용자와 그룹이 파일 및 디렉터리를 생성하고, 쓰고, 실행하고 다른 사용자는 새 파일과 디렉터리를 보거나 수정하거나 실행하지 못하게 하는 umask 설정이 있어야 합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade perms-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish perms-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

파일에 대한 액세스 제어

이 랩에서는 파일에 대한 권한을 구성하고, 특정 그룹의 사용자가 로컬 파일 시스템의 파일을 공유하는 데 사용할 수 있는 디렉터리를 설정합니다.

결과

- 사용자가 공동으로 파일 작업을 할 수 있는 디렉터리를 생성합니다.
- 그룹 소유권이 자동으로 할당되는 파일을 생성합니다.
- 그룹 외부에서 액세스할 수 없는 파일을 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start perms-review
```

지침

- student** 사용자로 **serverb**에 로그인합니다. 쉘 프롬프트에서 **sudo -i** 명령을 실행하여 **root** 사용자가 됩니다. **student**를 **student** 사용자 암호로 사용합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- /home/techdocs** 디렉터리를 생성합니다.

1. **mkdir** 명령을 사용하여 **/home/techdocs** 디렉터리를 생성합니다.

```
[root@serverb ~]# mkdir /home/techdocs
```

2. **/home/techdocs** 디렉터리의 그룹 소유권을 **techdocs** 그룹으로 변경합니다.

3. 3.1. **chown** 명령을 사용하여 **/home/techdocs** 디렉터리의 그룹 소유권을 **techdocs** 그룹으로 변경합니다.

```
[root@serverb ~]# chown :techdocs /home/techdocs
```

4. **techdocs** 그룹의 사용자가 **/home/techdocs** 디렉터리에 파일을 생성할 수 있는지 확인합니다.

- 4.1. **su** 명령을 사용하여 **tech1** 사용자로 전환합니다.

```
[root@serverb ~]# su - tech1  
[tech1@serverb ~]$
```

- 4.2. **/home/techdocs** 디렉터리에 **techdoc1.txt** 파일을 생성합니다. 이 단계는 실패합니다.

techdocs 그룹이 **/home/techdocs** 디렉터리를 소유하고 있고 **tech1**이 **techdocs** 그룹에 속하지만 해당 디렉터리에 파일을 생성할 수 없습니다. 그 이유는 **techdocs** 그룹에 쓰기 권한이 없기 때문입니다.

```
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt  
touch: cannot touch '/home/techdocs/techdoc1.txt': Permission denied
```

- 4.3. 디렉터리의 권한을 표시합니다.

```
[tech1@serverb ~]$ ls -ld /home/techdocs/  
drwxr-xr-x. 2 root techdocs 6 Feb 5 16:05 /home/techdocs/
```

5. **/home/techdocs** 디렉터리에 대한 권한을 설정합니다. **/home/techdocs** 디렉터리에서 **setgid(2)**, 소유자/사용자 및 그룹의 경우 읽기, 쓰기, 실행 권한(7), 기타 사용자의 경우 권한 없음(0)을 구성합니다.

- 5.1. **tech1** 사용자 쉘을 종료합니다.

```
[tech1@serverb ~]$ exit  
logout  
[root@serverb ~]#
```

- 5.2. **/home/techdocs** 디렉터리에 대한 그룹 권한을 설정합니다. **setgid**, 소유자 및 그룹의 경우 읽기, 쓰기, 실행 권한, 기타의 경우 권한 없음을 구성합니다.

```
[root@serverb ~]# chmod 2770 /home/techdocs
```

6. 권한이 정확하게 설정되었는지 확인합니다.

```
[root@serverb ~]# ls -ld /home/techdocs  
drwxrws--- 2 root techdocs 6 Feb 4 18:12 /home/techdocs/
```

이제 **techdocs** 그룹에 쓰기 권한이 있습니다.

7. 이제 **techdocs** 그룹의 사용자가 **/home/techdocs** 디렉터리에 파일을 생성하고 편집할 수 있는지 확인합니다. **techdocs** 그룹에 속하지 않은 사용자는 **/home/techdocs** 디렉터리에 파일을 생성하거나 편집할 수 없습니다. **tech1** 및 **tech2** 사용자는 **techdocs** 그룹에 속해 있습니다. **database1** 사용자는 해당 그룹에 속하지 않습니다.

- 7.1. **tech1** 사용자로 전환합니다. **/home/techdocs** 디렉터리에 **techdoc1.txt** 파일을 생성합니다. **/home/techdocs/techdoc1.txt** 파일에 몇 가지 텍스트를 추가합니다. **tech1** 사용자 쉘을 종료합니다.

```
[root@serverb ~]# su - tech1
[tech1@serverb ~]$ touch /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
-rw-r--r--. 1 tech1 techdocs 0 Feb  5 16:42 /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ echo "This is the first tech doc."
> /home/techdocs/techdoc1.txt
[tech1@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 7.2. **tech2** 사용자로 전환합니다. `/home/techdocs/techdoc1.txt` 파일의 콘텐츠를 표시합니다. `/home/techdocs` 디렉터리에 `techdoc2.txt` 파일을 생성합니다. **tech2** 사용자 쉘을 종료합니다.

```
[root@serverb ~]# su - tech2
[tech2@serverb ~]$ cd /home/techdocs
[tech2@serverb techdocs]$ cat techdoc1.txt
This is the first tech doc.
[tech2@serverb techdocs]$ touch /home/techdocs/techdoc2.txt
[tech2@serverb techdocs]$ ls -l
total 4
-rw-r--r--. 1 tech1 techdocs 28 Feb  5 17:43 techdoc1.txt
-rw-r--r--. 1 tech2 techdocs 0 Feb  5 17:45 techdoc2.txt
[tech2@serverb techdocs]$ exit
logout
[root@serverb ~]#
```

- 7.3. **database1** 사용자로 전환합니다. `/home/techdocs/techdoc1.txt` 파일의 콘텐츠를 표시합니다. **Permission Denied** 메시지가 표시됩니다. **database1** 사용자에 파일 액세스 권한이 있는지 확인합니다. **database1** 사용자 쉘을 종료합니다.

다음 긴 `echo` 명령을 한 줄에 입력합니다.

```
[root@serverb ~]# su - database1
[database1@serverb ~]$ cat /home/techdocs/techdoc1.txt
cat: /home/techdocs/techdoc1.txt: Permission denied
[database1@serverb ~]$ ls -l /home/techdocs/techdoc1.txt
ls: cannot access '/home/techdocs/techdoc1.txt': Permission denied
[database1@serverb ~]$ exit
logout
[root@serverb ~]#
```

8. `/etc/login.defs` 파일을 수정하여 로그인 쉘의 기본 umask를 조정합니다. 일반 사용자에게는 해당 사용자와 그룹이 파일 및 디렉터리를 생성하고, 쓰고, 실행하고 다른 사용자는 새 파일과 디렉터리를 보거나 수정하거나 실행하지 못하게 하는 umask 설정이 있어야 합니다.

- 8.1. **student** 사용자의 umask를 결정합니다. **student** 로그인 쉘로 전환합니다. 마쳤으면 쉘을 종료합니다.

```
[root@serverb ~]# su - student
[student@serverb ~]$ umask
0022
[student@serverb ~]$ exit
logout
[root@serverb ~]#
```

- 8.2. `/etc/login.defs` 파일을 편집하고 umask를 **007**로 설정합니다. `/etc/login.defs` 파일에는 이미 umask 정의가 포함되어 있습니다. 파일을 검색하고 적절한 값으로 업데이트합니다.

```
[root@serverb ~]# cat /etc/login.defs
...output omitted...
UMASK          007
...output omitted...
```

- 8.3. `student` 사용자로 전역 umask가 **007**로 변경되는지 확인합니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ umask
0007
```

- 8.4. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade perms-review
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish perms-review
```

이것으로 섹션을 완료합니다.

요약

- **chmod** 명령은 명령줄에서 파일 권한을 변경합니다.
- **chmod** 명령은 심볼릭 또는 8진수의 두 가지 방법 중 하나를 사용하여 권한을 나타낼 수 있습니다.
- **chown** 명령은 파일 소유권을 변경합니다. **chown** 명령의 **-R** 옵션은 디렉터리 트리의 소유권을 재귀적으로 변경합니다.
- 인수 없이 **umask** 명령을 실행하면 쉘의 현재 umask 값이 표시됩니다. 시스템의 모든 프로세스에는 umask가 있습니다.
- Bash의 기본 umask 값은 **/etc/login.defs** 파일에 정의되며, **/etc/profile** 및 **/etc/bashrc** 파일, **/etc/profile.d**의 파일 또는 사용자 계정의 쉘 초기화 파일의 설정으로 인해 영향을 받을 수 있습니다.
- **suid**, **sgid**, **sticky** 특수 권한은 파일에 대한 추가 액세스 관련 기능을 제공합니다.

5장

SELinux 보안 관리

목적

SELinux를 사용하여 서버의 보안을 보호하고 관리합니다.

목표

- SELinux에서 리소스를 보호하고, 시스템의 현재 SELinux 모드를 변경하고, 시스템의 기본 SELinux 모드를 설정하는 방법을 설명합니다.
- **semanage fcontext** 명령을 사용하여 파일 및 디렉터리의 기본 컨텍스트를 결정하고 **restorecon** 명령을 사용하여 SELinux 정책에서 정의한 컨텍스트를 파일 및 디렉터리에 적용하는 SELinux 정책 규칙을 관리합니다.
- **setsebool** 명령을 사용하여 SELinux 정책 규칙을 활성화 및 비활성화하고, **semanage boolean -l** 명령을 사용하여 SELinux 부울의 영구 값을 관리하고, **_selinux**로 끝나는 **man** 페이지를 참조하여 SELinux 부울에 대한 유용한 정보를 찾습니다.
- SELinux 로그 분석 툴을 사용하고 **sealert** 명령으로 SELinux 문제를 해결하는 동안 유용한 정보를 표시합니다.

섹션

- SELinux 적용 모드 변경(안내에 따른 연습)
- SELinux 파일 컨텍스트 제어(안내에 따른 연습)
- 부울을 사용하여 SELinux 정책 조정(안내에 따른 연습)
- SELinux 문제 조사 및 해결(안내에 따른 연습)

랩

SELinux 보안 관리

SELinux 적용 모드 변경

목표

SELinux에서 리소스를 보호하고, 시스템의 현재 SELinux 모드를 변경하고, 시스템의 기본 SELinux 모드를 설정하는 방법을 설명합니다.

SELinux 아키텍처

SELinux(Security Enhanced Linux)는 Linux의 중요한 보안 기능입니다. 파일, 포트 및 기타 리소스에 대한 액세스가 매우 세밀한 수준으로 제어됩니다. 프로세스는 해당 SELinux 정책 또는 SELinux 부울 설정에서 지정하는 리소스에만 액세스할 수 있습니다.

파일 권한은 특정 사용자 또는 그룹의 파일 액세스 권한을 제어합니다. 그러나 파일 권한은 파일 액세스 권한이 있는 승인된 사용자가 파일을 의도하지 않은 용도로 사용하는 것을 방지하지 못합니다.

예를 들어 파일에 대한 쓰기 권한이 있는 경우, 다른 편집기나 프로그램에서 특정 프로그램만 작성할 수 있도록 설계된 구조화된 데이터 파일을 여전히 열고 수정할 수 있으며, 이로 인해 손상 또는 데이터 보안 문제가 발생할 수 있습니다. 파일 권한은 파일 사용 방법은 제어하지 않고 파일을 읽거나 쓰거나 실행할 수 있는 사람만 제어하므로 이처럼 원하지 않는 액세스를 차단하지 않습니다.

SELinux는 애플리케이션 개발자가 애플리케이션에서 사용하는 각 바이너리 실행 파일, 구성 파일, 데이터 파일에 대해 허용되는 조치와 액세스 권한을 선언하기 위해 정의한 애플리케이션별 정책으로 구성됩니다. 하나의 정책이 애플리케이션 하나의 활동을 정의하기 때문에 이 정책을 타겟 정책이라고 합니다. 정책은 개별 프로그램, 파일, 네트워크 포트에 구성된 사전 정의 레이블을 선언합니다.

SELinux 사용

SELinux는 프로세스와 리소스 간에 허용된 작업을 명시적으로 정의하는 액세스 규칙 집합을 적용합니다. 액세스 규칙에 정의되어 있지 않은 작업은 허용되지 않습니다. 정의된 작업만 허용되므로 보안 설계가 취약한 애플리케이션도 악의적인 사용으로부터 보호됩니다. 타겟 정책이 있는 애플리케이션 또는 서비스는 제한된 도메인에서 실행되는 반면, 정책이 없는 애플리케이션은 제한 없이 실행되며 SELinux로 보호되지 않습니다. 개별 타겟 정책을 비활성화하여 애플리케이션과 보안 정책의 개발 및 디버깅을 지원할 수 있습니다.

SELinux에는 다음과 같은 작동 모드가 있습니다.

- **Enforcing(적용)**: SELinux가 로드된 정책을 적용합니다. 이 모드는 Red Hat Enterprise Linux의 기본값입니다.
- **Permissive(허용)**: SELinux가 정책을 로드하고 활성 상태지만 액세스 제어 규칙을 적용하지 않고 액세스 위반을 기록합니다. 이 모드는 애플리케이션 및 규칙을 테스트하고 문제를 해결하는데 유용합니다.
- **Disabled(비활성화됨)**: SELinux가 꺼져 있습니다. SELinux 위반이 거부되거나 기록되지 않습니다. SELinux는 비활성화하지 않는 것이 좋습니다.

**중요**

Red Hat Enterprise Linux 9부터는 부팅 시 **selinux=0** 커널 매개 변수를 사용해야만 SELinux를 완전히 비활성화할 수 있습니다. RHEL은 더 이상 **/etc/selinux/config** 파일에서 **SELINUX=disabled** 옵션 설정을 지원하지 않습니다.

RHEL 9부터는 **/etc/selinux/config** 파일에서 SELinux를 비활성화하면 SELinux에서 활성 상태의 적용을 시작하고 수행하지만 정책은 로드하지 않습니다. 정책 규칙은 허용된 작업을 정의하므로 정책이 로드되지 않으면 모든 작업이 거부됩니다. 이 동작은 의도된 것이며 SELinux 보호를 우회하려는 악의적인 시도를 차단하기 위해 설계되었습니다.

기본 SELinux 개념

SELinux의 기본 목적은 손상된 애플리케이션 또는 시스템 서비스로 인한 부적절한 사용으로부터 사용자 데이터를 보호하는 것입니다. 대부분의 Linux 관리자는 필요에 따라 파일 권한을 설정하기 때문에 DAC(Discretionary Access Control)라는 표준 사용자, 그룹, 월드 파일 권한 보안 모델에 익숙합니다. SELinux는 MAC(Mandatory Access Control)이라는 세분화된 규칙에 정의된 오브젝트 기반 보안 기능을 추가로 제공합니다. MAC 정책은 모든 사용자에게 적용되므로 임의의 구성 설정을 통해 특정 사용자에게 적용되지 않게 할 수 없습니다.

예를 들어 웹 서버의 개방형 방화벽 포트는 웹 클라이언트에 대한 원격 익명 액세스를 허용합니다. 그러나 해당 포트에 액세스하는 악의적인 사용자가 기존 취약점을 이용해 시스템을 손상시킬 수도 있습니다. 예를 들어 취약점이 **apache** 사용자 및 그룹의 권한을 손상시키는 경우, 악의적인 사용자가 **/var/www/html** 문서 루트 컨텐츠 또는 시스템의 **/tmp** 및 **/var/tmp** 디렉터리 또는 기타 액세스 가능한 파일 및 디렉터리에 직접 액세스할 수 있습니다.

SELinux 정책은 특정 프로세스가 관련 파일, 디렉터리, 포트에 액세스하는 방법을 정의하는 보안 규칙입니다. 파일, 프로세스, 디렉터리 또는 포트와 같은 모든 리소스 엔터티에는 SELinux 컨텍스트라는 레이블이 있습니다. 컨텍스트 레이블은 프로세스에서 레이블이 지정된 리소스에 액세스할 수 있도록 정의된 SELinux 정책 규칙을 찾습니다. 기본적으로 SELinux 정책은 명시적 규칙에서 액세스 권한을 부여하지 않는 한 어떠한 액세스도 허용하지 않습니다. 허용 규칙이 정의되지 않은 경우 모든 액세스가 허용되지 않습니다.

SELinux 레이블에는 **user**, **role**, **type**, **security level** 필드가 있습니다. 타겟 정책은 RHEL에서 기본적으로 활성화되어 있으며, **type** 컨텍스트를 사용하여 규칙을 정의합니다. 유형 컨텍스트 이름은 대개 **_t**로 끝납니다.

<i>SELinux User</i>	<i>Role</i>	<i>Type</i>	<i>Level</i>	<i>File</i>
unconfined_u:object_r:httdp_sys_content_t:s0				/var/www/html/file2

그림 5.1: SELinux 파일 컨텍스트

정책 액세스 규칙 개념

예를 들어 웹 서버 프로세스는 **httpd_t** 유형 컨텍스트로 레이블이 지정됩니다. **/var/www/html** 디렉터리 및 기타 위치의 웹 서버 파일과 디렉터리는 **httpd_sys_content_t** 유형 컨텍스트로 레이블이 지정됩니다. **/tmp** 및 **/var/tmp** 디렉터리의 임시 파일에는 **tmp_t** 유형 컨텍스트가 레이블로 사용됩니다. 웹 서버의 포트에는 레이블로 **http_port_t** 유형 컨텍스트가 사용됩니다.

Apache 웹 서버 프로세스는 **httpd_t** 유형 컨텍스트로 실행됩니다. 정책 규칙에서는 Apache 서버가 **httpd_sys_content_t** 유형 컨텍스트를 사용하여 레이블이 지정된 파일 및 디렉터리에 액세스하도록 허용합니다. 기본적으로 **/var/www/html** 디렉터리의 파일에는 **httpd_sys_content_t** 유형 컨텍스트가 있습니다. 웹 서버 정책에는 기본적으로 **/tmp** 및 **/var/tmp** 디렉터리와 같이 **tmp_t** 레이블이 지정된 파일을 사용하는 데 필요한 **allow** 규칙이 없으므로 액세스가 허용되지 않습니다. SELinux를 활성화해도

손상된 Apache 프로세스를 사용하는 악의적인 사용자는 `/tmp` 디렉터리 파일에 여전히 액세스할 수 없습니다.

MariaDB 서버 프로세스는 `mysqld_t` 유형 컨텍스트로 실행됩니다. 기본적으로 `/data/mysql` 디렉터리의 파일에는 `mysqld_db_t` 유형 컨텍스트가 있습니다. MariaDB 서버는 `mysqld_db_t` 레이블이 지정된 파일에는 액세스할 수 있지만, 기타 서비스 파일(예: `httpd_sys_content_t` 레이블이 지정된 파일)에 대한 액세스를 허용하는 규칙은 없습니다.

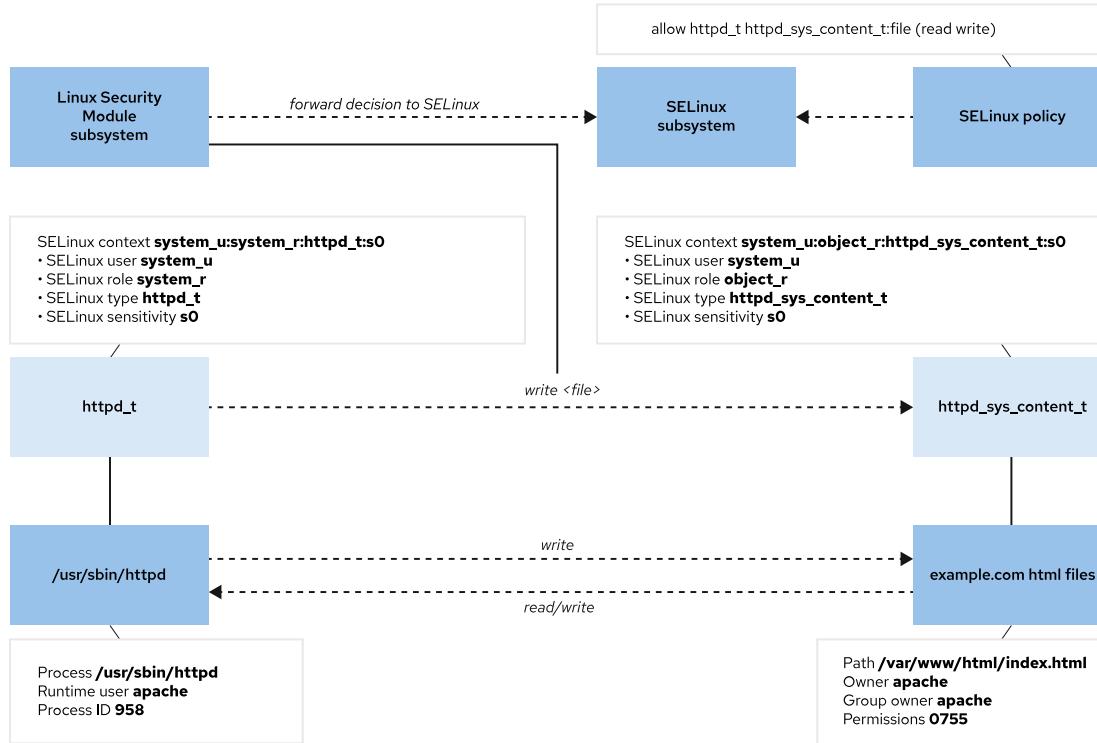


그림 5.2: SELinux의 의사 결정 흐름

리소스를 나열하는 대부분의 명령은 `-Z` 옵션을 사용하여 SELinux 컨텍스트를 관리합니다. 예를 들어 `ps`, `ls`, `cp`, `mkdir` 명령은 모두 `-Z` 옵션을 사용합니다.

```

[root@host ~]# ps axZ
LABEL PID TTY STAT TIME COMMAND
system_u:system_r:kernel_t:s0 2 ? S 0:00 [kthreadd]
system_u:system_r:kernel_t:s0 3 ? I< 0:00 [rcu_gp]
system_u:system_r:kernel_t:s0 4 ? I< 0:00 [rcu_par_gp]
...output omitted...
[root@host ~]# systemctl start httpd
[root@host ~]# ps -ZC httpd
LABEL PID TTY TIME CMD
system_u:system_r:httpd_t:s0 1550 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1551 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1552 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1553 ? 00:00:00 httpd
system_u:system_r:httpd_t:s0 1554 ? 00:00:00 httpd
[root@host ~]# ls -Z /var/www
system_u:object_r:httpd_sys_script_exec_t:s0 cgi-bin
system_u:object_r:httpd_sys_content_t:s0 html
  
```

SELinux 모드 변경

`getenforce` 명령을 사용하여 현재 SELinux 모드를 봅니다. `setenforce` 명령을 사용하여 SELinux 모드를 변경합니다.

```
[root@host ~]# getenforce
Enforcing
[root@host ~]# setenforce
usage: setenforce [ Enforcing | Permissive | 1 | 0 ]
[root@host ~]# setenforce 0
[root@host ~]# getenforce
Permissive
[root@host ~]# setenforce Enforcing
[root@host ~]# getenforce
Enforcing
```

또는 부팅 시 커널 매개 변수를 사용하여 SELinux 모드를 설정합니다. `enforcing=0` 커널 매개 변수를 전달하여 시스템을 `permissive` 모드로 부팅하거나 `enforcing=1`을 전달하여 `enforcing` 모드로 부팅합니다. `selinux=0` 커널 매개 변수를 전달하여 SELinux를 비활성화하거나 `selinux=1`을 전달하여 SELinux를 활성화합니다.

SELinux 모드를 `Permissive`에서 `Enforcing`으로 변경하는 경우 서버를 재부팅하는 것이 좋습니다. 이렇게 재부팅하면 허용 모드에서 시작된 서비스가 다음 부팅에서 제한됩니다.

기본 SELinux 모드 설정

SELinux를 영구적으로 구성하려면 `/etc/selinux/config` 파일을 사용합니다. 다음 기본 예제에서 구성은 SELinux를 `enforcing` 모드로 설정합니다. 주석에는 `permissive` 및 `disabled` 모드와 같은 기타 유효한 값이 나열됩니다.

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#       enforcing - SELinux security policy is enforced.
#       permissive - SELinux prints warnings instead of enforcing.
#       disabled - No SELinux policy is loaded.
...output omitted...
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#     grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#     grubby --update-kernel ALL --remove-args selinux
#
SELINUX=enforcing
# SELINUXTYPE= can take one of these three values:
#       targeted - Targeted processes are protected,
```

```
#      minimum - Modification of targeted policy. Only selected processes are
#      protected.
#      mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

시스템은 부팅 시 이 파일을 읽고 그에 따라 SELinux를 구성합니다. 커널 인수 **selinux=0|1** 및 **enforcing=0|1**은 이 구성을 재정의합니다.



참조

[getenforce\(8\)](#), [setenforce\(8\)](#) 및 [selinux_config\(5\)](#) 도움말 페이지

▶ 연습 가이드

SELinux 적용 모드 변경

이 랩에서는 SELinux 모드를 일시적 및 영구적으로 관리합니다.

결과

- 현재 SELinux 모드를 보고 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-opsmode
```

지침

- ▶ 1. **workstation** 시스템에서 **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 기본 SELinux 모드를 permissive(허용)로 변경합니다.

- 2.1. **getenforce** 명령을 사용하여 **servera** 시스템의 현재 SELinux 모드를 확인합니다.

```
[root@servera ~]# getenforce
Enforcing
```

- 2.2. **vim /etc/selinux/config** 명령을 사용하여 구성 파일을 편집합니다. **SELINUX** 매개 변수를 **enforcing**에서 **permissive** 모드로 변경합니다.

```
[root@servera ~]# vim /etc/selinux/config
```

- 2.3. **grep** 명령을 사용하여 **SELINUX** 매개 변수가 **permissive** 모드로 표시되는지 확인합니다.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=permissive
SELINUXTYPE=targeted
```

- 2.4. **setenforce** 명령을 사용하여 SELINUX 매개 변수를 **permissive** 모드로 변경하고 변경 사항을 확인합니다.

```
[root@servera ~]# setenforce 0
[root@servera ~]# getenforce
Permissive
```

- ▶ 3. 구성 파일에서 기본 SELinux 모드를 다시 **enforcing** 모드로 변경합니다.

- 3.1. **vim /etc/selinux/config** 명령을 사용하여 구성 파일을 편집합니다. SELINUX 매개 변수를 **permissive**에서 **enforcing** 모드로 변경합니다.

```
[root@servera ~]# vim /etc/selinux/config
```

- 3.2. **grep** 명령을 사용하여 부팅 시 SELINUX 매개 변수가 **enforcing** 모드로 설정되는지 확인합니다.

```
[root@servera ~]# grep '^SELINUX' /etc/selinux/config
SELINUX=enforcing
SELINUXTYPE=targeted
```

- ▶ 4. 명령줄에서 SELinux 모드를 **enforcing**으로 설정합니다. **servera** 시스템을 재부팅하고 SELinux 모드를 확인합니다.

- 4.1. **setenforce** 명령을 사용하여 현재 SELinux 모드를 **enforcing** 모드로 설정합니다. **getenforce** 명령을 사용하여 SELinux가 **enforcing** 모드로 설정되어 있는지 확인합니다.

```
[root@servera ~]# setenforce 1
[root@servera ~]# getenforce
Enforcing
```

- 4.2. **servera** 시스템을 재부팅하여 구성을 영구적으로 구현합니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 4.3. **servera** 시스템에 로그인하여 SELinux 모드를 확인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# getenforce
Enforcing
```

- ▶ 5. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-opsmode
```

이것으로 섹션을 완료합니다.

SELinux 파일 컨텍스트 제어

목표

`semanage fcontext` 명령을 사용하여 파일 및 디렉터리의 기본 컨텍스트를 결정하고, `restorecon` 명령을 사용하여 SELinux 정책에서 정의한 컨텍스트를 파일 및 디렉터리에 적용하는 SELinux 정책 규칙을 관리합니다.

초기 SELinux 컨텍스트

프로세스, 파일, 포트와 같은 모든 리소스에는 SELinux 컨텍스트로 레이블이 지정됩니다. SELinux는 `/etc/selinux/targeted-contexts/files/` 디렉터리에서 파일 레이블 지정 정책에 대한 파일 기반 데이터베이스를 유지 관리합니다. 새 파일은 해당 파일 이름이 기존 레이블 지정 정책과 일치하는 경우 기본 레이블을 가져옵니다.

새 파일의 이름이 기존 레이블 지정 정책과 일치하지 않는 경우에는 파일에 상위 디렉터리와 동일한 레이블이 상속됩니다. 레이블 상속을 사용하면 파일에 대한 명시적 정책이 존재하는지의 여부와 관계없이 모든 파일이 생성될 때 항상 레이블이 지정됩니다.

기존의 레이블 지정 정책이 있는 기본 위치에 파일이 생성되거나 사용자 지정 위치에 대한 정책이 있는 경우 새 파일에 올바른 SELinux 컨텍스트로 레이블이 지정됩니다. 그러나 기존 레이블 지정 정책 없이 파일이 예기치 않은 위치에 생성되면 상속된 레이블이 새 파일의 의도된 용도에 맞지 않을 수 있습니다.

또한 파일을 새 위치에 복사하면 해당 파일의 SELinux 컨텍스트가 새 위치의 레이블 지정 정책 또는 상위 디렉터리 상속(정책이 없는 경우)으로 결정된 새 컨텍스트로 변경될 수 있습니다. 복사하는 동안 파일의 SELinux 컨텍스트를 보존하여 파일의 원래 위치에 대해 결정된 컨텍스트 레이블을 유지할 수 있습니다. 예를 들어 `cp -p` 명령은 가능한 경우 모든 파일 특성을 유지하고 `cp --preserve=context` 명령은 복사하는 동안 SELinux 컨텍스트만 유지합니다.



참고

파일을 복사하면 항상 파일 inode가 생성되며 이전에 설명한 대로 SELinux 컨텍스트를 포함하여 해당 inode의 속성을 처음에 설정해야 합니다.

그러나 파일을 동일한 파일 시스템 내에서 이동하는 경우에는 이동해도 inode가 생성되지 않고, 대신 기존 inode의 파일 이름이 새 위치로 이동합니다. 기존 inode의 속성을 초기화할 필요가 없으므로 `-Z` 옵션을 사용하여 파일에 새 컨텍스트를 설정하지 않는 한 `mv`를 사용하여 이동한 파일은 SELinux 컨텍스트가 유지됩니다.

파일을 복사하거나 이동한 후에는 적절한 SELinux 컨텍스트가 있는지 확인하고 필요한 경우 올바르게 설정합니다.

다음 예제는 이 프로세스의 작동 방식을 보여줍니다.

`/tmp` 디렉터리에 두 개의 빈 파일을 생성합니다. 두 파일 모두 `user_tmp_t` 컨텍스트 유형을 수신합니다.

첫 번째 파일을 이동하고 두 번째 파일을 `/var/www/html` 디렉터리에 복사합니다.

- 이동된 파일에는 원래 `/tmp` 디렉터리에서 레이블이 지정된 파일 컨텍스트가 유지됩니다.

5장 | SELinux 보안 관리

- 복사된 파일에는 새 inode가 있으며 대상 `/var/www/html` 디렉터리에서 SELinux 컨텍스트를 상속합니다.

`ls -Z` 명령은 파일의 SELinux 컨텍스트를 표시합니다. `/tmp` 디렉터리에 생성된 파일의 레이블을 살펴봅니다.

```
[root@host ~]# touch /tmp/file1 /tmp/file2
[root@host ~]# ls -Z /tmp/file*
unconfined_u:object_r:user_tmp_t:s0 /tmp/file1
unconfined_u:object_r:user_tmp_t:s0 /tmp/file2
```

`ls -Zd` 명령은 지정된 디렉터리의 SELinux 컨텍스트를 표시합니다. `/var/www/html` 디렉터리의 레이블과 그 안의 파일을 확인합니다.

```
[root@host ~]# ls -Zd /var/www/html/
system_u:object_r:httpd_sys_content_t:s0 /var/www/html/
[root@host ~]# ls -Z /var/www/html/index.html
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/index.html
```

하나의 파일을 `/tmp` 디렉터리에서 `/var/www/html` 디렉터리로 이동합니다. 기타 파일을 동일한 디렉터리에 복사합니다. 각 파일의 결과 레이블을 확인합니다.

```
[root@host ~]# mv /tmp/file1 /var/www/html/
[root@host ~]# cp /tmp/file2 /var/www/html/
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

이동된 파일은 원래 레이블을 유지하고 복사된 파일은 대상 디렉터리 레이블을 상속합니다. `unconfined_u` 는 SELinux 사용자 역할이고, `object_r` 은 SELinux 역할이며, `s0` 는 (가능한 가장 낮은) 민감도 수준입니다. 고급 SELinux 구성 및 기능에서는 이러한 값을 사용합니다.

SELinux 컨텍스트 변경

파일의 SELinux 컨텍스트는 `semanage fcontext`, `restorecon`, `chcon` 명령으로 관리할 수 있습니다.

파일의 컨텍스트를 변경하는 데 권장되는 방법은 `semanage fcontext` 명령을 사용하여 파일 컨텍스트 정책을 생성한 다음 `restorecon` 명령을 사용하여 정책에 지정된 컨텍스트를 파일에 적용하는 것입니다. 이 방법을 사용하면 필요할 때마다 `restorecon` 명령을 사용하여 파일의 레이블을 올바른 컨텍스트로 다시 지정할 수 있습니다. 이 방법의 장점은 컨텍스트가 무엇인지 기억할 필요가 없으며 파일 집합에서 컨텍스트를 수정할 수 있다는 것입니다.

`chcon` 명령은 SELinux 컨텍스트를 파일에서 직접 변경하지만 시스템의 SELinux 정책은 참조하지 않습니다. `chcon` 은 테스트 및 디버깅에 유용하지만 이 방법을 사용하여 컨텍스트를 수동으로 변경하는 것은 일시적입니다. 수동으로 변경할 수 있는 파일 컨텍스트는 재부팅 후에도 유지되지만 `restorecon` 을 실행하여 파일 시스템 콘텐츠에 레이블을 다시 지정하는 경우 바뀔 수 있습니다.

**중요**

SELinux 시스템 레이블 재지정이 발생하면 시스템의 모든 파일에 해당 정책 기본값으로 레이블이 지정됩니다. 파일에 **restorecon**을 사용하는 경우 SELinux 정책의 규칙과 일치하지 않으면 파일에서 수동으로 변경한 컨텍스트가 대체됩니다.

다음 예제에서는 / 상위 디렉터리에서 상속된 **default_t** SELinux 컨텍스트를 사용하여 디렉터리를 생성합니다.

```
[root@host ~]# mkdir /virtual
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

chcon 명령은 **/virtual** 디렉터리의 파일 컨텍스트를 **httpd_sys_content_t** 유형으로 설정합니다.

```
[root@host ~]# chcon -t httpd_sys_content_t /virtual
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:httpd_sys_content_t:s0 /virtual
```

restorecon 명령을 실행하면 컨텍스트가 기본값인 **default_t**로 재설정됩니다. **Relabeled** 메시지를 확인합니다.

```
[root@host ~]# restorecon -v /virtual
Relabeled /virtual from unconfined_u:object_r:httpd_sys_content_t:s0 to
unconfined_u:object_r:default_t:s0
[root@host ~]# ls -Zd /virtual
unconfined_u:object_r:default_t:s0 /virtual
```

SELinux 기본 파일 컨텍스트 정책 정의

semanage fcontext 명령은 기본 파일 컨텍스트를 결정하는 정책을 표시하고 수정합니다. **semanage fcontext -l** 명령을 실행하여 모든 파일 컨텍스트 정책 규칙을 나열할 수 있습니다. 이러한 규칙은 확장된 정규 표현식 구문을 사용하여 경로 및 파일 이름을 지정합니다.

정책을 볼 때 가장 일반적인 확장 정규 표현식은 **(/.*)?**이며, 일반적으로 디렉터리 이름에 추가됩니다. 이 표기법은 유머러스하게 해적이라고 합니다. 얼굴에 눈 가리개가 있고 그 옆에 갈고리 손이 있는 것처럼 보이기 때문입니다.

이 구문은 '슬래시로 시작하고 그 뒤에 임의의 수의 문자가 있는 문자 집합'으로 설명되며, 집합은 있을 수도 있고 없을 수도 있습니다. 더 간단히 말하면 이 구문은 비어 있는 경우에도 디렉터리 자체와 일치하지만 해당 디렉터리 내에 생성된 거의 모든 파일 이름과 일치합니다.

예를 들어 다음 규칙은 더 구체적인 규칙으로 재정의하지 않는 한, **/var/www/cgi-bin** 디렉터리와 이 디렉터리 또는 하위 디렉터리(및 해당 하위 디렉터리 등)에 있는 모든 파일에 **system_u:object_r:httpd_sys_script_exec_t:s0** SELinux 컨텍스트가 있어야 함을 지정합니다.

```
/var/www/cgi-bin(/.*)? all files system_u:object_r:httpd_sys_script_exec_t:s0
```

**참고**

위 예의 **all files** 필드 옵션은 지정하지 않은 경우 **semanage**에서 사용하는 기본 파일 형식입니다. 이 옵션은 **semanage**와 함께 사용할 수 있는 모든 파일 유형에 적용되며, Red Hat System Administration I (RH124) 교육 과정의 파일에 대한 액세스 제어 장에 표시된 표준 파일 유형과 동일합니다. **semanage-fcontext(8)** 도움말 페이지에서 추가 정보를 얻을 수 있습니다.

기본 파일 컨텍스트 작업

다음 표는 SELinux 파일 컨텍스트 정책을 추가, 제거 또는 나열하는 **semanage fcontext** 명령에 대한 참조입니다.

semanage fcontext 명령

옵션	설명
-a, --add	지정한 개체 유형의 기록 추가
-d, --delete	지정한 개체 유형의 기록 삭제
-l, --list	지정한 개체 유형의 기록 나열

SELinux 컨텍스트를 관리하려면 **restorecon** 및 **semanage** 명령이 포함된 **policycoreutils** 및 **policycoreutils-python-utils** 패키지를 설치하십시오.

디렉터리의 모든 파일을 기본 정책 컨텍스트로 재설정하려면 먼저 **semanage fcontext -l** 명령을 사용하여 올바른 정책을 찾고 의도한 파일 컨텍스트에 있는지 확인합니다. 그런 다음 와일드카드 디렉터리 이름에 **restorecon** 명령을 사용하여 모든 파일을 재귀적으로 재설정합니다. 다음 예제에서 **semanage** 및 **restorecon** 명령을 사용하기 전과 이후에 파일 컨텍스트를 확인합니다.

먼저 SELinux 컨텍스트에서 파일을 확인합니다.

```
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:user_tmp_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

그런 다음 **semanage fcontext -l** 명령을 사용하여 기본 SELinux 파일 컨텍스트를 나열합니다.

```
[root@host ~]# semanage fcontext -l
...output omitted...
/var/www/(.*)?    all files    system_u:object_r:httpd_sys_content_t:s0
...output omitted...
```

semanage 명령 출력은 **/var/www/** 디렉터리의 모든 파일과 하위 디렉터리에 기본적으로 **httpd_sys_content_t** 컨텍스트가 있음을 나타냅니다. 와일드카드 디렉터리에서 **restorecon** 명령을 실행하면 모든 파일 및 하위 디렉터리에서 기본 컨텍스트가 복원됩니다.

```
[root@host ~]# restorecon -Rv /var/www/
Relabeled /var/www/html/file1 from unconfined_u:object_r:user_tmp_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Z /var/www/html/file*
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file1
unconfined_u:object_r:httpd_sys_content_t:s0 /var/www/html/file2
```

다음 예제에서는 **semanage** 명령을 사용하여 새 디렉터리에 대한 컨텍스트 정책을 추가합니다. 먼저 **index.html** 파일이 포함된 **/virtual** 디렉터리를 생성합니다. 파일 및 디렉터리에 대한 SELinux 컨텍스트를 확인합니다.

```
[root@host ~]# mkdir /virtual
[root@host ~]# touch /virtual/index.html
[root@host ~]# ls -Zd /virtual/
unconfined_u:object_r:default_t:s0 /virtual
[root@host ~]# ls -Z /virtual/
unconfined_u:object_r:default_t:s0 index.html
```

다음으로 **semanage fcontext** 명령을 사용하여 디렉터리에 대한 SELinux 파일 컨텍스트 정책을 추가합니다.

```
[root@host ~]# semanage fcontext -a -t httpd_sys_content_t '/virtual(/.*)?'
```

와일드카드 디렉터리에서 **restorecon** 명령을 사용하여 디렉터리 및 디렉터리 내의 모든 파일에 대한 기본 컨텍스트를 설정합니다.

```
[root@host ~]# restorecon -RFvv /virtual
Relabeled /virtual from unconfined_u:object_r:default_t:s0 to
system_u:object_r:httpd_sys_content_t:s0
Relabeled /virtual/index.html from unconfined_u:object_r:default_t:s0 to
system_u:object_r:httpd_sys_content_t:s0
[root@host ~]# ls -Zd /virtual/
drwxr-xr-x. root root system_u:object_r:httpd_sys_content_t:s0 /virtual/
[root@host ~]# ls -Z /virtual/
-rw-r--r--. root root system_u:object_r:httpd_sys_content_t:s0 index.html
```

기본 정책에 대한 로컬 사용자 지정을 보려면 **semanage fcontext -l -C** 명령을 사용합니다.

```
[root@host ~]# semanage fcontext -l -C
SELinux fcontext      type          Context
/virtual(/.*)?        all files    system_u:object_r:httpd_sys_content_t:s0
```



참조

[chcon\(1\)](#), [restorecon\(8\)](#), [semanage\(8\)](#) 및 [semanage-fcontext\(8\)](#) 도움말 페이지

▶ 연습 가이드

SELinux 파일 컨텍스트 제어

이 랩에서는 디렉터리 및 디렉터리 내용의 SELinux 컨텍스트를 영구적으로 변경합니다.

결과

- 비표준 문서 루트에서 웹 콘텐츠를 게시하도록 Apache HTTP 서버를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-filecontexts
```

지침

- ▶ 1. **servera**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 비표준 위치에서 문서 디렉터리를 사용하도록 Apache를 구성합니다.

- 2.1. **/custom** 디렉터리를 생성합니다.

```
[root@servera ~]# mkdir /custom
```

- 2.2. **This is SERVERA.** 텍스트가 포함된 **/custom** 디렉터리에 **index.html** 파일을 만듭니다.

```
[root@servera ~]# echo 'This is SERVERA.' > /custom/index.html
```

- 2.3. 새 디렉터리 위치를 사용하도록 Apache를 구성합니다. Apache **/etc/httpd/conf/httpd.conf** 구성 파일을 편집하여 **/var/www/html** 디렉터리 두 개를 **/custom** 디렉터리로 바꿉니다. **vim /etc/httpd/conf/httpd.conf** 명령을 사용하여 이 작업을 수행 할 수 있습니다. 다음 예제에서는 예상되는 **/etc/httpd/conf/httpd.conf** 파일 내용을 보여줍니다.

```
[root@servera ~]# cat /etc/httpd/conf/httpd.conf
...output omitted...
DocumentRoot "/custom"
...output omitted...
<Directory "/custom">
...output omitted...
```

- ▶ 3. Apache 웹 서비스를 시작 및 활성화하고 서비스가 실행 중인지 확인합니다.

3.1. `systemctl` 명령을 사용하여 Apache 웹 서비스를 시작하고 활성화합니다.

```
[root@servera ~]# systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/
lib/systemd/system/httpd.service.
```

3.2. 서비스가 실행 중인지 확인합니다.

```
[root@servera ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
  preset: disabled)
    Active: active (running) since Wed 2022-04-06 05:21:19 EDT; 22s ago
      Docs: man:httpd.service(8)
     Main PID: 1676 (httpd)
...output omitted...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Starting The Apache HTTP
  Server...
Apr 06 05:21:19 servera.lab.example.com systemd[1]: Started The Apache HTTP
  Server.
Apr 06 05:21:19 servera.lab.example.com httpd[1676]: Server configured, listening
  on: port 80
```

- ▶ 4. `workstation`에서 웹 브라우저를 열고 `http://servera/index.html` 웹 페이지를 확인합니다. 파일에 액세스할 수 있는 권한이 없다는 오류 메시지가 표시됩니다.
- ▶ 5. `servera`의 `index.html` 파일에 대한 액세스를 허용하려면 SELinux 컨텍스트를 구성해야 합니다. `/custom` 디렉터리 및 디렉터리 아래의 모든 파일에 대한 컨텍스트 유형을 `httpd_sys_content_t`로 설정하는 SELinux 파일 컨텍스트 규칙을 정의합니다.

```
[root@servera ~]# semanage fcontext -a \
-t httpd_sys_content_t '/custom(/.*)?'
```

- ▶ 6. `/custom` 디렉터리의 파일 컨텍스트를 수정합니다.

```
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

▶ 7. **workstation** 시스템의 웹 브라우저에서 **http://servera/index.html**을 다시 확인합니다. **This is SERVERA.** 메시지가 표시되어야 합니다.

▶ 8. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-filecontexts
```

이것으로 섹션을 완료합니다.

부울을 사용하여 SELinux 정책 조정

목표

`setsebool` 명령을 사용하여 SELinux 정책 규칙을 활성화 및 비활성화하고, `semanage boolean -l` 명령을 사용하여 SELinux 부울의 영구 값을 관리하고, `_selinux`로 끝나는 `man` 페이지를 참조하여 SELinux 부울에 대한 유용한 정보를 찾습니다.

SELinux 부울

애플리케이션 또는 서비스 개발자는 SELinux 타겟 정책을 작성하여 타겟 애플리케이션의 허용된 동작을 정의합니다. 개발자는 특정 시스템에서 동작이 허용되는 경우 활성화할 수 있는 선택적 애플리케이션 동작을 SELinux 정책에 포함할 수 있습니다. SELinux 부울은 SELinux 정책의 선택적 동작을 활성화하거나 비활성화합니다. 부울을 사용하면 애플리케이션 동작을 선택적으로 조정할 수 있습니다.

이러한 선택적 동작은 애플리케이션에 따라 다르며 타겟 애플리케이션별로 검색하여 선택해야 합니다. 서비스별 부울은 해당 서비스의 SELinux 도움말 페이지에 설명되어 있습니다. 예를 들어 웹 서버 `httpd` 서비스에는 해당 `httpd(8)` 도움말 페이지와 지원되는 프로세스 유형과 파일 컨텍스트 및 사용 가능한 부울이 활성화된 동작을 포함하여 SELinux 정책을 문서화하는 `httpd_selinux(8)` 도움말 페이지가 있습니다. SELinux 도움말 페이지는 `selinux-policy-doc` 패키지에 제공됩니다.

`getsebool` 명령을 사용하여 이 시스템의 타겟 정책에 사용할 수 있는 부울과 현재 부울 상태를 나열합니다. `setsebool` 명령을 사용하여 이러한 동작의 실행 상태를 활성화하거나 비활성화합니다. `setsebool -P` 명령 옵션은 설정을 정책 파일에 작성하여 영구적으로 만듭니다. 권한 있는 사용자만 SELinux 부울을 설정할 수 있습니다.

```
[root@host ~]# getsebool -a
abrt_anon_write --> off
abrt_handle_event --> off
abrt_upload_watch_anon_write --> on
...output omitted...
```

httpd 정책 부울의 예

`httpd` 서비스 정책에는 `httpd`와 홈 디렉터리를 공유할 수 있는 `httpd_enable_homedirs` 부울이 포함됩니다. 일반적으로 사용자의 로컬 홈 디렉터리는 로컬 시스템에 로그인한 사용자만 액세스할 수 있습니다. 또는 NFS와 같은 원격 파일 공유 프로토콜을 사용하여 홈 디렉터리를 공유하고 홈 디렉터리에 액세스합니다. 두 시나리오 모두 기본적으로 `https`를 사용하여 홈 디렉터리를 공유하거나 사용자가 브라우저를 통해 사용할 수 없습니다.

```
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> off
```

공유를 활성화하고 사용자가 브라우저를 사용하여 홈 디렉터리에 액세스하도록 설정할 수 있습니다. 이렇게 설정한 경우 `httpd` 서비스는 `user_home_dir_t` 파일 컨텍스트로 레이블이 지정된 홈 디렉터리를 공유합니다. 그러면 사용자가 브라우저에서 홈 디렉터리 파일에 액세스하여 파일을 관리할 수 있습니다.

정책 부울 관리

-P 옵션 없이 **semanage boolean** 명령을 사용하여 SELinux 부울을 설정하는 것은 일시적이며 해당 설정은 재부팅 후 영구 값으로 돌아갑니다. **semanage boolean -l** 명령을 사용하여 추가 정보를 확인합니다. 이 명령은 정책 파일의 부울과 함께 부울이 영구적인지의 여부, 기본값과 현재 값, 간단한 설명을 표시합니다.

```
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (off , off)  Allow httpd to enable homedirs
[root@host ~]# setsebool httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs          (on , off)  Allow httpd to enable homedirs
[root@host ~]# getsebool httpd_enable_homedirs
httpd_enable_homedirs --> on
```

현재 설정이 부팅 시 기본 설정과 다른 부울만 나열하려면 **semanage boolean -l -C** 명령을 사용합니다. 이 예제는 **grep** 필터링 없이도 위 예제와 동일한 결과를 제공합니다.

```
[root@host ~]# semanage boolean -l -C
SELinux boolean           State  Default Description
httpd_enable_homedirs      (on ,   off)  Allow httpd to enable homedirs
```

이전 예제에서는 시스템이 재부팅될 때까지 **httpd_enable_homedirs** 부울의 현재 값을 임시로 **on**으로 설정했습니다. 기본 설정을 변경하려면 **setsebool -P** 명령을 사용하여 설정을 영구적으로 만듭니다. 다음 예제에서는 영구 값을 설정한 다음 정책 파일에서 부울 정보를 봅니다.

```
[root@host ~]# setsebool -P httpd_enable_homedirs on
[root@host ~]# semanage boolean -l | grep httpd_enable_homedirs
httpd_enable_homedirs      (on ,   on)  Allow httpd to enable homedirs
```

semanage boolean -l -C 명령을 다시 사용합니다. 현재 설정과 기본 설정이 동일한 것처럼 보여도 부울이 표시됩니다. 그러나 현재 설정이 마지막 부팅의 기본 설정과 다른 경우 -C 옵션이 일치합니다. 이 **httpd_enable_homedirs** 예제의 경우 원래 기본 부팅 설정은 **off**입니다.

```
[root@host ~]# semanage boolean -l -C
SELinux boolean           State  Default Description
httpd_enable_homedirs      (on ,   on)  Allow httpd to enable homedirs
```



참조

booleans(8), **getsebool(8)**, **setsebool(8)**, **semanage(8)** 및 **semanage-boolean(8)** 도움말 페이지

▶ 연습 가이드

부울을 사용하여 SELinux 정책 조정

이 예제에서는 사용자의 홈 디렉터리에 있는 웹 콘텐츠를 게시하도록 Apache를 구성합니다.

결과

- 사용자의 홈 디렉터리에 있는 웹 콘텐츠를 게시하도록 Apache 웹 서비스를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-booleans
```

지침

- ▶ 1. **workstation** 시스템에서 **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **/etc/httpd/conf.d/userdir.conf** 구성 파일을 편집하여 사용자가 홈 디렉터리에 있는 웹 콘텐츠를 게시할 수 있도록 Apache 기능을 활성화합니다. **IfModule** 섹션에서 **UserDir** 변수를 **disabled** 값으로 설정하는 행을 주석 처리하고 **UserDir** 변수를 **public_html** 값으로 설정하는 행의 주석 처리를 해제합니다.

```
[root@servera ~]# vim /etc/httpd/conf.d/userdir.conf
<IfModule mod_userdir.c>
...output omitted...
# UserDir disabled

...output omitted...
UserDir public_html

...output omitted...
</IfModule>
```

- ▶ 3. Apache 웹 서비스를 시작하고 활성화합니다.

```
[root@servera ~]# systemctl enable --now httpd
```

- ▶ 4. 다른 터미널 창을 열고 **ssh** 명령을 사용하여 **servera** 시스템에 **student** 사용자로 로그인합니다. **~/public_html** 디렉터리에 **index.html** 웹 콘텐츠 파일을 생성합니다.

- 4.1. 다른 터미널 창에서 **ssh** 명령을 사용하여 **servera** 시스템에 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 4.2. **mkdir** 명령을 사용하여 **~/public_html** 디렉터리를 생성합니다.

```
[student@servera ~]$ mkdir ~/public_html
```

- 4.3. 다음 내용으로 **index.html** 파일을 생성합니다.

```
[student@servera ~]$ echo 'This is student content on SERVERA.' > \
~/public_html/index.html
```

- 4.4. Apache 웹 서비스에서 **/home/student/public_html** 디렉터리의 콘텐츠를 제공하려면 **/home/student** 디렉터리의 파일과 하위 디렉터리를 공유할 수 있어야 합니다. **/home/student/public_html** 디렉터리를 생성할 때 홈 디렉터리 권한이 있는 모든 사용자가 해당 콘텐츠에 액세스할 수 있는 권한이 자동으로 구성됩니다.

Apache 웹 서비스가 **public_html** 하위 디렉터리에 액세스할 수 있도록 **/home/student** 디렉터리 권한을 변경합니다.

```
[student@servera ~]$ chmod 711 ~
[student@servera ~]$ ls -ld ~
drwx--x--x. 16 student student 4096 Nov  3 09:28 /home/student
```

- ▶ 5. **workstation** 시스템에서 웹 브라우저를 열고 주소 **http://servera/~student/index.html**을 입력합니다. 파일에 액세스할 수 있는 권한이 없다는 오류 메시지가 표시됩니다.

- ▶ 6. 다른 터미널로 전환하고 **getsebool** 명령을 사용하여 **httpd** 서비스의 홈 디렉터리에 대한 액세스를 제한하는 부울이 있는지 확인합니다.

```
[root@servera ~]# getsebool -a | grep home
...output omitted...
httpd_enable_homedirs --> off
...output omitted...
```

- ▶ 7. **setsebool** 명령을 사용하여 **httpd** 서비스의 홈 디렉터리에 영구적으로 액세스할 수 있도록 설정합니다.

```
[root@servera ~]# setsebool -P httpd_enable_homedirs on
```

- ▶ 8. 이제 주소 `http://servera/~student/index.html`을 입력한 후 웹 브라우저에 **This is student content on SERVERA.** 메시지가 표시되는지 확인합니다. 메시지를 보기 위해 웹 브라우저를 닫았다가 다시 열어야 할 수 있습니다.
- ▶ 9. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-booleans
```

이것으로 섹션을 완료합니다.

SELinux 문제 조사 및 해결

목표

SELinux 로그 분석 툴을 사용하고 **sealert** 명령으로 SELinux 문제를 해결하는 동안 유용한 정보를 표시합니다.

SELinux 문제 해결

SELinux 액세스 거부로 인해 애플리케이션이 예기치 않게 작동하지 않는 경우 이러한 문제를 해결하는 방법과 툴을 사용할 수 있습니다. SELinux가 활성화되어 있는 경우 먼저 몇 가지 기본 개념과 동작을 이해하는 것이 좋습니다.

- SELinux는 허용되는 작업을 명시적으로 정의하는 타겟 정책으로 구성됩니다.
- 정책 항목은 상호 작용할 레이블이 지정된 프로세스와 리소스를 정의합니다.
- 정책은 레이블을 사용하여 프로세스 유형과 파일 또는 포트 컨텍스트를 나타냅니다.
- 정책 항목은 하나의 프로세스 유형, 하나의 리소스 레이블, 허용할 명시적 작업을 정의합니다.
- 작업은 시스템 호출, 커널 함수 또는 다른 특정 프로그래밍 루틴이 될 수 있습니다.
- 특정 프로세스, 리소스, 작업 간 관계를 나타내는 항목이 생성되지 않은 경우 작업이 거부됩니다.
- 작업이 거부되면 유용한 컨텍스트 정보와 함께 시도가 기록됩니다.

Red Hat Enterprise Linux는 배포판의 거의 모든 서비스에 대해 안정적인 타겟 SELinux 정책을 제공합니다. 따라서 올바르게 구성된 경우 일반적인 RHEL 서비스에서 SELinux 액세스 문제가 발생하는 것은 드문 경우입니다. SELinux 액세스 문제는 서비스가 잘못 구현되거나 새 애플리케이션에 불완전한 정책이 있는 경우 발생합니다. SELinux 구성을 광범위하게 변경하기 전에 이러한 문제 해결 개념을 고려하십시오.

- 대부분의 액세스 거부는 부적절한 작업을 차단하여 SELinux가 정확하게 작동하고 있음을 나타냅니다.
- 거부된 작업을 평가하려면 일반적인 예상 서비스 작업에 어느 정도 익숙해야 합니다.
- 가장 일반적인 SELinux 문제는 새로 생성하거나 복사 또는 이동한 파일의 컨텍스트가 올바르지 않은 경우입니다.
- 파일 컨텍스트는 기존 정책이 해당 위치를 참조할 때 수정됩니다.
- 선택적 부울 정책 기능은 **_selinux** 도움말 페이지에 설명되어 있습니다.
- 부울 기능을 구현하려면 일반적으로 비 SELinux 구성을 추가로 설정해야 합니다.
- SELinux 정책은 파일 권한 또는 액세스 제어 목록 제한을 대체하거나 우회하지 않습니다.

일반 애플리케이션 또는 서비스가 실패하고 서비스에 작동 중인 SELinux 정책이 있는 것으로 확인되면 먼저 서비스의 **_selinux** 도움말 페이지를 확인하여 컨텍스트 유형 레이블이 올바른지 확인합니다. 영향 받는 프로세스 및 파일 속성을 보고 레이블이 올바르게 설정되어 있는지 확인합니다.

SELinux 위반 모니터링

setroubleshoot-server 패키지의 SELinux 문제 해결 서비스에서는 SELinux 문제를 진단하는 툴을 제공합니다. SELinux에서 작업을 거부하면 **/var/log/audit/audit.log** 보안 로그 파일에 AVC(Access Vector Cache) 메시지가 기록됩니다. SELinux 문제 해결 서비스는 AVC 이벤트를 모니터링하고 이벤트 요약을 **/var/log/messages** 파일로 전송합니다.

AVC 요약에는 이벤트 고유의 식별자(UUID)가 포함됩니다. 특정 이벤트에 대한 포괄적인 보고서 세부 정보를 보려면 **sealert -l UUID** 명령을 사용합니다. 기존 이벤트를 모두 보려면 **sealert -a /var/log/audit/audit.log** 명령을 사용합니다.

표준 Apache 웹 서버의 다음과 같은 예제 명령 시퀀스를 고려하십시오. `/root/mypage` 를 생성하여 기본 Apache 콘텐츠 디렉터리(`/var/www/html`)로 이동합니다. 그런 다음 Apache 서비스를 시작한 후 파일 콘텐츠를 검색합니다.

```
[root@host ~]# touch /root/mypage
[root@host ~]# mv /root/mypage /var/www/html
[root@host ~]# systemctl start httpd
[root@host ~]# curl http://localhost/mypage
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access this resource.</p>
</body></html>
```

웹 서버에서 콘텐츠를 표시하지 않고 `permission denied` 오류를 반환합니다. AVC 이벤트는 `/var/log/audit/audit.log` 및 `/var/log/messages` 파일에 기록됩니다. `/var/log/messages` 이벤트 메시지에서 제안된 `sealert` 명령 및 UUID를 확인합니다.

```
[root@host ~]# tail /var/log/audit/audit.log
...output omitted...
type=AVC msg=audit(1649249057.067:212): avc: denied { getattr } for pid=2332 comm="httpd" path="/var/www/html/mypage" dev="vda4" ino=9322502 scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0
...output omitted
[root@host ~]# tail /var/log/messages
...output omitted...
Apr  6 08:44:19 host setroubleshoot[2547]: SELinux is preventing /usr/sbin/httpd from getattr access on the file /var/www/html/mypage. For complete SELinux messages run: sealert -l 95f41f98-6b56-45bc-95da-ce67ec9a9ab7
...output omitted...
```

`sealert` 출력에는 이벤트가 설명되어 있으며 영향받는 프로세스, 액세스한 파일, 시도 및 거부된 작업 등이 포함되어 있습니다. 출력에는 해당하는 경우 파일의 레이블을 수정하는데 필요한 조언이 포함됩니다. 추가 조언에서는 거부된 작업을 허용하는 새 정책 생성 방법을 설명합니다. 시나리오에 적합한 경우에만 제공된 조언을 사용하십시오.



중요

`sealert` 출력에는 지정된 조언으로 거부를 완화할 신뢰 수준을 나타내는 신뢰도 등급이 포함됩니다. 그러나 해당 조언은 사용자의 시나리오에 적합하지 않을 수 있습니다.

예를 들어 거부된 파일이 잘못된 위치에 있기 때문에 AVC가 거부된 경우 파일의 컨텍스트 레이블을 조정하거나 이 위치 및 작업에 대해 정책을 생성하라는 조언은 기술적으로는 정확하지만 해당 시나리오에는 올바르지 않은 솔루션입니다. 근본 원인이 잘못된 위치 또는 파일 이름인 경우 파일을 이동하거나 파일 이름을 변경한 다음 올바른 파일 컨텍스트를 복원하는 것이 올바른 솔루션입니다.

```
[root@host ~]# sealert -l 95f41f98-6b56-45bc-95da-ce67ec9a9ab7
SELinux is preventing /usr/sbin/httpd from getattr access on the file /var/www/
html/mypage.

***** Plugin restorecon (99.5 confidence) suggests ****
If you want to fix the label.
/var/www/html/mypage default label should be httpd_sys_content_t.
Then you can run restorecon. The access attempt may have been stopped due to
insufficient permissions to access a parent directory in which case try to change
the following command accordingly.
Do
# /sbin/restorecon -v /var/www/html/mypage

***** Plugin catchall (1.49 confidence) suggests ****
If you believe that httpd should be allowed getattr access on the mypage file by
default.
Then you should report this as a bug.
You can generate a local policy module to allow this access.
Do
allow this access for now by executing:
#ausearch -c 'httpd' --raw | audit2allow -M my-httpd
# semodule -X 300 -i my-httpd.pp

Additional Information:
Source Context          system_u:system_r:httpd_t:s0
Target Context          unconfined_u:object_r:admin_home_t:s0
Target Objects          /var/www/html/mypage [ file ]
Source                 httpd
Source Path             /usr/sbin/httpd
...output omitted...

Raw Audit Messages
type=AVC msg=audit(1649249057.67:212): avc: denied { getattr }
for pid=2332 comm="httpd" path="/var/www/html/mypage"
dev="vda4" ino=9322502 scontext=system_u:system_r:httpd_t:s0
tcontext=unconfined_u:object_r:admin_home_t:s0 tclass=file permissive=0

type=SYSCALL msg=audit(1649249057.67:212): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7fe9c00048f8 a2=7fe9ccfc8830 a3=100
items=0 ppid=2329 pid=2332 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48
egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm=httpd exe=/usr/sbin/httpd
subj=system_u:system_r:httpd_t:s0 key=(null)

Hash: httpd,httpd_t,admin_home_t,file,getattr
```

이 예제에서 액세스된 파일은 올바른 위치에 있지만 올바른 SELinux 파일 컨텍스트가 없습니다. **Raw Audit Messages** 섹션에는 `/var/log/audit/audit.log` 이벤트 항목의 정보가 표시됩니다. `restorecon /var/www/html/mypage` 명령을 사용하여 올바른 컨텍스트 레이블을 설정합니다. 여러 파일을 재귀적으로 수정하려면 상위 디렉터리에서 `restorecon -R` 명령을 사용합니다.

/var/log/audit/audit.log 명령을 사용하여 ausearch 로그 파일에서 AVC 이벤트를 검색합니다. -m 옵션을 사용하여 AVC 메시지 유형을 지정하고 -ts 옵션을 사용하여 recent와 같은 시간 힌트를 제공합니다.

```
[root@host ~]# ausearch -m AVC -ts recent
-----
time->Tue Apr  6 13:13:07 2019
type=PROCTITLE msg=audit(1554808387.778:4002):
  proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1554808387.778:4002): arch=c000003e syscall=49
  success=no exit=-13 a0=3 a1=55620b8c9280 a2=10 a3=7ffed967661c items=0
  ppid=1 pid=9340 auid=4294967295 uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0
  sgid=0 fsgid=0 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
  subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1554808387.778:4002): avc: denied { name_bind } for pid=9340 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0 tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
```

웹 콘솔을 사용하여 SELinux 문제 해결

RHEL 웹 콘솔에는 SELinux 문제를 해결하는 툴이 포함되어 있습니다. 왼쪽 메뉴에서 SELinux를 선택합니다. SELinux Policy(SELinux 정책) 창에 현재 적용 상태가 표시됩니다. SELinux access control errors 섹션에는 현재 SELinux 문제가 나열됩니다.

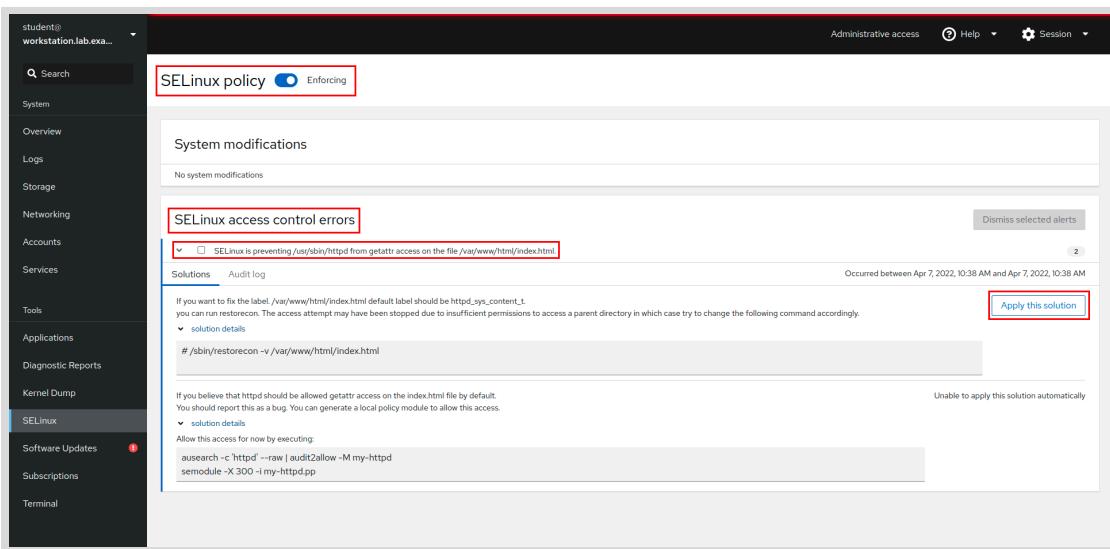


그림 5.3: 웹 콘솔의 SELinux 정책 및 오류

> 문자를 클릭하여 이벤트 세부 정보를 표시합니다. **solution details**를 클릭하여 모든 이벤트 세부 정보 및 조언을 표시합니다. **Apply the solution**을 클릭할 수 있습니다.

문제를 수정한 후 SELinux access control errors 섹션은 해당 이벤트를 보기에서 제거해야 합니다. No SELinux alerts 메시지가 나타나면 현재 SELinux 문제를 모두 수정한 것입니다.



참조

[sealert\(8\)](#) 도움말 페이지

▶ 연습 가이드

SELinux 문제 조사 및 해결

이 랩에서는 SELinux 보안 거부 문제를 해결하는 방법을 학습합니다.

결과

- SELinux 문제 해결 툴을 사용해 봅니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-issues
```

지침

- ▶ 1. **workstation** 시스템의 웹 브라우저에서 **http://servera/index.html** 웹 페이지를 엽니다. 파일에 액세스할 수 있는 권한이 없다는 오류 메시지가 표시됩니다.
- ▶ 2. **ssh** 명령을 사용하여 **servera**에 **student** 사용자로 로그인합니다. **sudo -i** 명령을 사용하여 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 3. **less** 명령을 사용하여 **/var/log/messages** 파일의 콘텐츠를 확인합니다. / 문자를 사용하고 **sealert** 텍스트를 검색합니다. 위 연습에서도 SELinux 메시지가 생성되었을 수 있으므로 마지막 항목에 도달할 때까지 **n** 키를 누릅니다. 다음 단계에서 사용할 수 있도록 제안된 **sealert** 명령을 복사합니다. **q** 명령을 종료하려면 **less** 키를 사용합니다.

```
[root@servera ~]# less /var/log/messages
...output omitted...
Apr  7 04:52:18 servera setroubleshoot[20715]: SELinux is preventing /usr/sbin/
httpd from getattr access on the file /custom/index.html. For complete SELinux
messages run: sealert -l 9a96294a-239b-4568-8f1e-9f35b5fb472b
...output omitted...
```

- ▶ 4. 제안된 **sealert** 명령을 실행합니다. 소스 컨텍스트, 대상 오브젝트, 정책 및 적용 모드를 확인합니다. **httpd** 서비스에서 제공할 파일의 올바른 SELinux 컨텍스트 레이블을 찾습니다.

- 4.1. **sealert** 명령을 실행합니다.

출력에 `/custom/index.html` 파일의 컨텍스트 레이블이 잘못되었다고 설명되어 있습니다.

```
[root@servera ~]# sealert -l 9a96294a-239b-4568-8f1e-9f35b5fb472b
SELinux is preventing /usr/sbin/httpd from getattr access on the file /custom/
index.html.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow httpd to have getattr access on the index.html file
Then you need to change the label on /custom/index.html

Do
# semanage fcontext -a -t FILE_TYPE '/custom/index.html'
where FILE_TYPE is one of the following: NetworkManager_exec_t,
NetworkManager_log_t, NetworkManager_tmp_t, abrt_dump_oops_exec_t,
abrt_etc_t, abrt_exec_t, abrt_handle_event_exec_t, abrt_helper_exec_t,
abrt_retrace_coredump_exec_t, abrt_retrace_spool_t, abrt_retrace_worker_exec_t,
abrt_tmp_t, abrt_upload_watch_tmp_t, abrt_var_cache_t, abrt_var_log_t,
abrt_var_run_t, accountsd_exec_t, acct_data_t, acct_exec_t, admin_crontab_tmp_t,
admin_passwd_exec_t, afs_logfile_t, aide_exec_t, aide_log_t, alsa_exec_t,
alsa_tmp_t, amanda_exec_t, amanda_log_t, amanda_recover_exec_t, amanda_tmp_t,
amtu_exec_t, anacron_exec_t, anon_inodefs_t
...output omitted...

Additional Information:
Source Context          system_u:system_r:httpd_t:s0
Target Context          unconfined_u:object_r:default_t:s0
Target Objects           /custom/index.html [ file ]
Source                 httpd
Source Path             /usr/sbin/httpd
Port                   <Unknown>
Host                   servera.lab.example.com
Source RPM Packages    httpd-2.4.51-7.el9_0.x86_64
Target RPM Packages    selinux-policy-targeted-34.1.27-1.el9.noarch
SELinux Policy RPM     selinux-policy-targeted-34.1.27-1.el9.noarch
Local Policy RPM        True
Selinux Enabled         targeted
Policy Type            Enforcing
Enforcing Mode         Enforcing
Host Name              servera.lab.example.com
Platform               Linux servera.lab.example.com
                        5.14.0-70.2.1.el9_0.x86_64 #1 SMP PREEMPT Wed Mar
                        16 18:15:38 EDT 2022 x86_64 x86_64
Alert Count             4
First Seen              2022-04-07 04:51:38 EDT
Last Seen               2022-04-07 04:52:13 EDT
Local ID                9a96294a-239b-4568-8f1e-9f35b5fb472b
```

```
Raw Audit Messages
type=AVC msg=audit(1649321533.406:1024): avc: denied { getattr } for
pid=20464 comm="httpd" path="/custom/index.html" dev="vda4" ino=25571802
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0

...output omitted...
```

- 4.2. SELinux 컨텍스트에 **httpd** 서비스에서 기본적으로 제공하는 콘텐츠가 포함된 디렉터리인 **/var/www/html**이 있는지 확인합니다. **httpd_sys_content_t** SELinux 컨텍스트는 **/custom/index.html** 파일에 적합합니다.

```
[root@servera ~]# ls -ldz /var/www/html
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 6 Mar 21 11:47 /
var/www/html
```

- ▶ 5. **sealert** 명령의 **Raw Audit Messages** 섹션에는 **/var/log/audit/audit.log** 파일의 정보가 포함됩니다. **/var/log/audit/audit.log** 명령을 사용하여 **ausearch** 파일을 검색합니다. **-m** 옵션은 메시지 유형을 검색합니다. **-ts** 옵션은 시간을 기준으로 검색합니다. 다음 항목은 경고를 유발하는 관련 프로세스 및 파일을 식별합니다. 프로세스는 **httpd** Apache 웹 서버이고, 파일은 **/custom/index.html**이며 컨텍스트는 **system_r:httpd_t**입니다.

```
[root@servera ~]# ausearch -m AVC -ts today
...output omitted...
-----
time->Thu Apr 7 04:52:13 2022
type=PROCTITLE msg=audit(1649321533.406:1024):
    proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1649321533.406:1024): arch=c000003e syscall=262 success=no
    exit=-13 a0=ffffffff9c a1=7fefc403d850 a2=7fefc89bc830 a3=100 items=0 ppid=20461
    pid=20464 auid=4294967295 uid=48 gid=48 euid=48 suid=48 egid=48
    sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
    subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1649321533.406:1024): avc: denied
    { getattr } for pid=20464 comm="httpd" path="/custom/index.html"
    dev="vda4" ino=25571802 scontext=system_u:system_r:httpd_t:s0
    tcontext=unconfined_u:object_r:default_t:s0 tclass=file permissive=0
```

- ▶ 6. **httpd_sys_content_t** 컨텍스트를 적용하여 문제를 해결합니다.

```
[root@servera ~]# semanage fcontext -a \
-t httpd_sys_content_t '/custom(/.*)?'
[root@servera ~]# restorecon -Rv /custom
Relabeled /custom from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
Relabeled /custom/index.html from unconfined_u:object_r:default_t:s0 to
unconfined_u:object_r:httpd_sys_content_t:s0
```

- ▶ 7. 다시 **http://servera/index.html**을 봅니다. **This is SERVERA.** 메시지가 표시됩니다.
▶ 8. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-issues
```

이것으로 섹션을 완료합니다.

▶ 랩

SELinux 보안 관리

이 랩에서는 시스템 로그 파일의 문제를 식별하고 SELinux 구성을 조정합니다.

결과

- 시스템 로그 파일의 문제를 확인합니다.
- SELinux 구성을 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-review
```

지침

1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.
2. **workstation** 시스템의 웹 브라우저에서 **http://serverb/lab.html** 웹 페이지를 확인합니다. 다음과 같은 오류 메시지가 표시됩니다. **You do not have permission to access this resource.**
3. Apache 서비스에서 웹 콘텐츠를 제공하지 못하도록 하는 SELinux 문제를 조사하고 식별합니다.
4. 새 HTTP 문서 디렉터리와 원본 HTTP 문서 디렉터리의 SELinux 컨텍스트를 표시합니다. Apache 서버에서 웹 콘텐츠를 제공하지 못하도록 하는 SELinux 문제를 해결합니다.
5. Apache 서버에서 이제 웹 콘텐츠를 제공할 수 있는지 확인합니다.
6. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade selinux-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

SELinux 보안 관리

이 랩에서는 시스템 로그 파일의 문제를 식별하고 SELinux 구성을 조정합니다.

결과

- 시스템 로그 파일의 문제를 확인합니다.
- SELinux 구성을 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start selinux-review
```

지침

1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

2. **workstation** 시스템의 웹 브라우저에서 <http://serverb/lab.html> 웹 페이지를 확인합니다. 다음과 같은 오류 메시지가 표시됩니다. **You do not have permission to access this resource.**
3. Apache 서비스에서 웹 콘텐츠를 제공하지 못하도록 하는 SELinux 문제를 조사하고 식별합니다.
 - 3.1. **/var/log/messages** 파일의 콘텐츠를 봅니다. **/** 키를 사용하고 **sealert** 문자열을 검색합니다. **q** 명령을 종료하려면 **less** 키를 사용합니다.

```
[root@serverb ~]# less /var/log/messages
...output omitted...
Apr  7 06:16:15 serverb setroubleshoot[26509]: failed to retrieve rpm info for /
lab-content/la
b.html
Apr  7 06:16:17 serverb setroubleshoot[26509]: SELinux is preventing /usr/sbin/
httpd from getattr access on the file /lab-content/lab.html. For complete SELinux
messages run: sealert -l 35c9e452-2552-4ca3-8217-493b72ba6d0b
Apr  7 06:16:17 serverb setroubleshoot[26509]: SELinux is preventing /usr/sbin/
httpd from getattr access on the file /lab-content/lab.html
...output omitted...
```

- 3.2. 제안된 **sealert** 명령을 실행합니다. 소스 컨텍스트, 대상 오브젝트, 정책 및 적용 모드를 확인합니다.

```
[root@serverb ~]# sealert -l 35c9e452-2552-4ca3-8217-493b72ba6d0b
SELinux is preventing /usr/sbin/httpd from getattr access on the file /lab-
content/lab.html.

***** Plugin catchall_labels (83.8 confidence) suggests *****

If you want to allow httpd to have getattr access on the lab.html file
Then you need to change the label on /lab-content/lab.html
Do
# semanage fcontext -a -t FILE_TYPE '/lab-content/lab.html'
where FILE_TYPE is one of the following:
...output omitted...

Additional Information:
Source Context          system_u:system_r:httpd_t:s0
Target Context          unconfined_u:object_r:default_t:s0
Target Objects          /lab-content/lab.html [ file ]
Source                 httpd
Source Path             /usr/sbin/httpd
Port                  <Unknown>
Host                  serverb.lab.example.com
Source RPM Packages    httpd-2.4.51-7.el9_0.x86_64
Target RPM Packages
SELinux Policy RPM     selinux-policy-targeted-34.1.27-1.el9.noarch
Local Policy RPM       selinux-policy-targeted-34.1.27-1.el9.noarch
Selinux Enabled         True
Policy Type            targeted
Enforcing Mode         Enforcing
Host Name              serverb.lab.example.com
Platform               Linux serverb.lab.example.com
                        5.14.0-70.2.1.el9_0.x86_64 #1 SMP PREEMPT Wed Mar
                        16 18:15:38 EDT 2022 x86_64 x86_64
Alert Count             8
First Seen              2022-04-07 06:14:45 EDT
Last Seen               2022-04-07 06:16:12 EDT
Local ID                35c9e452-2552-4ca3-8217-493b72ba6d0b

Raw Audit Messages
```

```

type=AVC msg=audit(1649326572.86:407): avc: denied { getattr } for
pid=10731 comm="httpd" path="/lab-content/lab.html" dev="vda4" ino=18192752
scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
tclass=file permissive=0

type=SYSCALL msg=audit(1649326572.86:407): arch=x86_64 syscall=newfstatat
success=no exit=EACCES a0=fffffff9c a1=7f7c8c0457c0 a2=7f7c887f7830 a3=100 items=0
ppid=10641 pid=10731 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48
egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm=httpd exe=/usr/sbin/httpd
subj=system_u:system_r:httpd_t:s0 key=(null)

Hash: httpd,httpd_t,default_t,file,getattr

```

- 3.3. **sealert** 명령의 **Raw Audit Messages** 섹션에는 **/var/log/audit/audit.log** 파일의 정보가 포함됩니다. **/var/log/audit/audit.log** 파일을 검색합니다. **-m** 옵션은 메시지 유형을 검색합니다. **-ts** 옵션은 시간을 기준으로 검색합니다. 다음 항목은 경고를 유발하는 관련 프로세스 및 파일을 식별합니다. 프로세스는 **httpd** Apache 웹 서버이고, 파일은 **/lab-content/lab.html**이며 컨텍스트는 **system_r:httpd_t**입니다.

```

[root@serverb ~]# ausearch -m AVC -ts recent
...output omitted...
-----
time->Thu Apr  7 06:16:12 2022
type=PROCTITLE msg=audit(1649326572.086:407):
  proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
type=SYSCALL msg=audit(1649326572.086:407): arch=c000003e syscall=262 success=no
  exit=-13 a0=fffffff9c a1=7f7c8c0457c0 a2=7f7c887f7830 a3=100 items=0 ppid=10641
  pid=10731 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48
  sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd"
  subj=system_u:system_r:httpd_t:s0 key=(null)
type=AVC msg=audit(1649326572.086:407): avc: denied { getattr } for
  pid=10731 comm="httpd" path="/lab-content/lab.html" dev="vda4" ino=18192752
  scontext=system_u:system_r:httpd_t:s0 tcontext=unconfined_u:object_r:default_t:s0
  tclass=file permissive=0

```

4. 새 HTTP 문서 디렉터리와 원본 HTTP 문서 디렉터리의 SELinux 컨텍스트를 표시합니다. Apache 서버에서 웹 콘텐츠를 제공하지 못하도록 하는 SELinux 문제를 해결합니다.

- 4.1. **/lab-content** 및 **/var/www/html** 디렉터리의 SELinux 컨텍스트를 비교합니다.

```

[root@serverb ~]# ls -dZ /lab-content /var/www/html
  unconfined_u:object_r:default_t:s0 /lab-content
  system_u:object_r:httpd_sys_content_t:s0 /var/www/html

```

- 4.2. **/lab-content** 디렉터리 및 그 아래의 모든 파일에 대한 기본 유형을 **httpd_sys_content_t**로 설정하는 파일 컨텍스트 규칙을 생성합니다.

```

[root@serverb ~]# semanage fcontext -a \
-t httpd_sys_content_t '/lab-content(/.*)?'

```

- 4.3. **/lab-content** 디렉터리의 파일에 대한 SELinux 컨텍스트를 수정합니다.

```
[root@serverb ~]# restorecon -R /lab-content/
```

5. Apache 서버에서 이제 웹 콘텐츠를 제공할 수 있는지 확인합니다.

- 5.1. 웹 브라우저를 사용하여 **http://serverb/lab.html** 링크를 새로 고칩니다. 콘텐츠가 표시되면 문제가 해결된 것입니다.

```
This is the html file for the SELinux final lab on SERVERB.
```

6. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade selinux-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish selinux-review
```

이것으로 섹션을 완료합니다.

요약

- **getenforce** 및 **setenforce** 명령은 시스템의 SELinux 모드를 관리하는데 사용됩니다.
- **semanage** 명령은 SELinux 정책 규칙을 관리합니다. **restorecon** 명령은 정책이 정의하는 컨텍스트를 적용합니다.
- 부울은 SELinux 정책의 동작을 변경하는 스위치입니다. 이를 활성화하거나 비활성화하여 정책을 조정할 수 있습니다.
- **sealert** 명령은 SELinux 문제 해결에 유용한 정보를 표시합니다.

시스템 성능 튜닝

목적

Red Hat Enterprise Linux 시스템에서 프로세스를 평가 및 제어하고, 튜닝 매개 변수를 설정하며, 프로세스 스케줄링 우선 순위를 조정합니다.

목표

- 명령을 사용하여 프로세스를 강제 종료 및 통신하고 데몬 프로세스의 특성을 정의하고, 사용자 세션 및 프로세스를 중지합니다.
- 부하 평균을 정의하고 리소스를 많이 사용하는 서버 프로세스를 확인합니다.
- 튜닝된 데몬이 관리하는 튜닝 프로파일을 선택하여 시스템 성능을 최적화합니다.
- nice 및 renice 명령을 사용하여 특정 프로세스에 우선 순위를 지정하거나 해제합니다.

섹션

- 프로세스 강제 종료(안내에 따른 연습)
- 프로세스 활동 모니터링(안내에 따른 연습)
- 튜닝 프로파일 조정(안내에 따른 연습)
- 프로세스 예약에 미치는 영향(안내에 따른 연습)

랩

- 시스템 성능 튜닝

프로세스 강제 종료

목표

명령을 사용하여 프로세스를 강제 종료 및 통신하고 데몬 프로세스의 특성을 정의하고, 사용자 세션 및 프로세스를 중지합니다.

신호를 사용한 프로세스 제어

신호는 프로세스에 전달된 소프트웨어 인터럽트입니다. 신호는 실행 중인 프로그램에 이벤트를 보고합니다. 신호를 생성하는 이벤트는 오류나 외부 이벤트(I/O 요청 또는 타이머 만료)일 수도 있고, 신호 전송 명령 또는 키보드 시퀀스의 명시적 사용에 의해 발생할 수도 있습니다.

다음 표에는 시스템 관리자가 일상적인 프로세스 관리에 사용하는 기본 신호가 나와 있습니다. 짧은 이름(HUP) 또는 올바른 이름(SIGHUP)으로 신호를 참조하십시오.

기본 프로세스 관리 신호

신호	이름	정의
1	HUP	Hangup : 터미널의 제어 프로세스 종료를 보고합니다. 또한 프로세스를 종료하지 않고 재초기화(구성 재로드)하도록 요청합니다.
2	INT	Keyboard interrupt : 프로그램이 종료됩니다. 차단하거나 처리할 수 있습니다. INTR(인터럽트) 키 시퀀스(Ctrl+c)를 눌러 전송합니다.
3	QUIT	Keyboard quit : SIGINT와 유사하며 종료할 때 프로세스 덤프를 추가합니다. QUIT 키 시퀀스(Ctrl+\)를 눌러 전송합니다.
9	KILL	Kill, unblockable : 갑작스럽게 프로그램이 종료됩니다. 차단, 무시 또는 처리할 수 없습니다. 일관성 있게 치명적입니다.
15 기본값	TERM	Terminate : 프로그램이 종료됩니다. SIGKILL과 달리 차단, 무시 또는 처리할 수 있습니다. 프로그램이 종료되도록 요청하는 "깔끔한" 방법입니다. 프로그램이 종료되기 전에 필수 작업 및 자체 정리를 완료할 수 있습니다.
18	CONT	Continue : 프로세스가 중지된 경우 재개하기 위해 전송합니다. 차단할 수 없습니다. 처리되더라도 항상 프로세스를 재개합니다.
19	STOP(중지)	Stop, unblockable : 프로세스를 일시 중지합니다. 차단하거나 처리할 수 없습니다.
20	TSTP	Keyboard stop : SIGSTOP와 달리 차단, 무시 또는 처리할 수 있습니다. 일시 중단 키 시퀀스(Ctrl+z)를 눌러 전송합니다.



참고

신호 번호는 Linux 하드웨어 플랫폼에 따라 다르지만, 신호 이름 및 의미는 표준입니다. 신호를 보낼 때 번호 대신 신호 이름을 사용하는 것이 좋습니다. 이 섹션에서 설명하는 번호는 x86_64 아키텍처 시스템에 해당합니다.

각 신호에는 기본 동작이 있으며, 일반적으로 다음 동작 중 하나입니다.

- **Term** : 즉시 하나의 프로그램을 종료합니다.
- **Core** : 프로그램의 메모리 이미지(코어 덤프)를 저장한 후 종료합니다.
- **Stop** : 실행 중인 프로그램을 중지(일시 중단)하고 계속(재개)되기를 기다립니다.

프로그램은 신호의 기본 동작을 무시, 교체 또는 확장하도록 핸들러 루틴을 구현하여 예상 이벤트 신호에 대응합니다.

명시적 요청을 통해 신호 전송

키보드 제어 시퀀스를 눌러 현재 포그라운드 프로세스에 프로세스를 일시 중단(**Ctrl+z**), 강제 종료 (**Ctrl+c**) 또는 코어 덤프(**Ctrl+**)하라는 신호를 보낼 수 있습니다. 그러나 신호 전송 명령을 사용하여 다른 세션의 백그라운드 프로세스에 신호를 보낼 수도 있습니다.

이름(예: **-HUP** 또는 **-SIGHUP** 옵션 사용)이나 번호(관련 **-1** 옵션 사용)로 신호를 지정할 수 있습니다. 사용자는 자신의 프로세스를 강제 종료할 수 있지만 다른 사용자가 소유한 프로세스를 강제 종료하려면 root 권한이 필요합니다.

kill 명령은 PID 번호를 사용하여 프로세스에 신호를 보냅니다. 이름과는 달리 **kill** 명령을 사용하면 프로그램 종료 신호뿐 아니라 모든 신호를 보낼 수 있습니다. **kill** 명령의 **-l** 옵션을 사용하여 사용 가능한 모든 신호의 이름과 번호를 표시할 수 있습니다.

```
[user@host ~]$ kill -l
 1) SIGHUP      2) SIGINT      3) SIGQUIT      4) SIGILL      5) SIGTRAP
 6) SIGABRT     7) SIGBUS      8) SIGFPE       9) SIGKILL     10) SIGUSR1
11) SIGSEGV     12) SIGUSR2     13) SIGPIPE     14) SIGALRM     15) SIGTERM
16) SIGSTKFLT   17) SIGCHLD     18) SIGCONT     19) SIGSTOP     20) SIGTSTP
...output omitted...
[user@host ~]$ ps aux | grep job
5194 0.0 0.1 222448 2980 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job1
5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job2
5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/user/bin/control job3
5430 0.0 0.0 221860 1096 pts/1 S+ 16:41 0:00 grep --color=auto job
[user@host ~]$ kill 5194
[user@host ~]$ ps aux | grep job
user 5199 0.0 0.1 222448 3132 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job2
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job3
user 5783 0.0 0.0 221860 964 pts/1 S+ 16:43 0:00 grep --color=auto
job
[1] Terminated control job1
[user@host ~]$ kill -9 5199
[user@host ~]$ ps aux | grep job
user 5205 0.0 0.1 222448 3124 pts/1 S 16:39 0:00 /bin/bash /home/
user/bin/control job3
user 5930 0.0 0.0 221860 1048 pts/1 S+ 16:44 0:00 grep --color=auto
job
[2]- Killed control job2
[user@host ~]$ kill -SIGTERM 5205
user 5986 0.0 0.0 221860 1048 pts/1 S+ 16:45 0:00 grep --color=auto job
[3]+ Terminated control job3
```

특정 프로세스 제어

pkill 명령을 사용하여 선택 기준에 맞는 하나 이상의 프로세스에 신호를 보냅니다. 선택 기준은 명령 이름, 특정 사용자가 소유한 프로세스 또는 모든 시스템 차원 프로세스일 수 있습니다.

프로세스 및 세션 신호는 개별적으로 또는 집합적으로 전송될 수 있습니다. 한 사용자의 모든 프로세스를 종료하려면 **pkill** 명령을 사용합니다.

로그인 세션의 초기 프로세스(세션 리더)는 세션 종료 요청을 처리하고 의도하지 않은 키보드 신호를 무시하도록 설계되었으므로 사용자 프로세스 및 로그인 쉘을 모두 강제 종료하려면 SIGKILL 신호가 필요합니다.

먼저 **pgrep** 명령을 사용하여 강제 종료할 PID 번호를 확인합니다. 이 명령은 대부분의 동일한 옵션을 포함하여 **pkill** 명령과 유사하게 작동합니다. 단, **pgrep** 명령은 프로세스를 강제 종료하지 않고 표시합니다.

pgrep 명령에 **-l** 옵션을 사용하면 프로세스 이름과 해당 ID가 나열됩니다. 두 명령 중 하나를 **-u** 옵션과 함께 사용하여 프로세스를 소유한 사용자의 ID를 지정합니다.

```
[root@host ~]# pgrep -l -u bob
6964 bash
6998 sleep
6999 sleep
7000 sleep
[root@host ~]# pkill -SIGKILL -u bob
[root@host ~]# pgrep -l -u bob
```

주의를 요하는 프로세스가 동일한 로그인 세션에 있는 경우에는 사용자의 모든 프로세스를 강제 종료할 필요가 없을 수도 있습니다. **w** 명령을 사용하여 세션의 제어 터미널을 확인한 다음, 동일한 터미널 ID를 참조하는 프로세스만 강제 종료합니다.

SIGKILL이 지정되지 않은 경우, 세션 리더(여기서는 Bash 로그인 쉘)는 종료 요청을 성공적으로 처리하여 생존하지만 다른 세션 프로세스는 모두 종료합니다.

-t 옵션을 사용하여 프로세스를 특정 터미널 ID와 일치시킵니다.

```
[root@host ~]# pgrep -l -u bob
7391 bash
7426 sleep
7427 sleep
7428 sleep
[root@host ~]# w -u bob
USER      TTY      FROM          LOGIN@    IDLE      JCPU      PCPU WHAT
bob        tty3                 18:37      5:04     0.03s   0.03s  -bash
[root@host ~]# pkill -t tty3
[root@host ~]# pgrep -l -u bob
7391 bash
[root@host ~]# pkill -SIGKILL -t tty3
[root@host ~]# pgrep -l -u bob
[root@host ~]#
```

**중요**

관리자는 일반적으로 SIGKILL을 사용합니다.

SIGKILL 신호는 처리하거나 무시할 수 없으므로 항상 치명적입니다. 그러나 이는 종료될 프로세스가 자체 클린업 루틴을 실행하도록 허용하지 않고 강제로 종료합니다. 먼저 SIGTERM 을 전송한 다음, SIGINT를 시도하고 둘 다 실패하는 경우에만 SIGKILL로 다시 시도하는 것이 좋습니다.

상위 및 하위 프로세스 관계를 사용하여 동일한 선택적 프로세스 종료를 적용할 수 있습니다. 시스템 또는 단일 사용자의 프로세스 트리를 보려면 **pstree** 명령을 사용합니다. 상위 프로세스가 생성한 모든 하위 프로세스를 강제 종료하려면 상위 프로세스의 PID를 사용합니다. 이번에는 신호가 하위 프로세스에서만 전달되었으므로 상위 **Bash** 로그인 쉘은 생존합니다.

```
[root@host ~]# pstree -p bob
bash(8391)---sleep(8425)
           ├─sleep(8426)
           └─sleep(8427)
[root@host ~]# pkill -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
[root@host ~]# pkill -SIGKILL -P 8391
[root@host ~]# pgrep -l -u bob
bash(8391)
```

여러 프로세스에 신호 보내기

killall 명령은 명령 이름을 기반으로 여러 프로세스에 신호를 보낼 수 있습니다.

```
[user@host ~]$ ps aux | grep job
5194  0.0  0.1 222448  2980 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job1
5199  0.0  0.1 222448  3132 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job2
5205  0.0  0.1 222448  3124 pts/1      S     16:39   0:00 /bin/bash /home/user/bin/
control job3
5430  0.0  0.0 221860  1096 pts/1      S+    16:41   0:00 grep --color=auto job
[user@host ~]$ killall control
[1]  Terminated                  control job1
[2]- Terminated                  control job2
[3]+ Terminated                  control job3
[user@host ~]$
```

백그라운드 작업 종료

백그라운드 작업을 종료하려면 **kill** 명령을 사용하고 작업 번호를 지정합니다.

jobs 명령을 사용하여 종료할 프로세스의 작업 번호를 찾습니다.

```
[user@host ~]$ jobs
[1]-  Running                  sleep 500 &
[2]+  Running                  sleep 1000 &
[user@host ~]$
```

kill 명령을 사용하여 특정 작업을 종료합니다. 작업 번호 앞에 퍼센트 기호(%)를 붙입니다.

```
[user@host ~]$ kill -SIGTERM %1
[user@host ~]$ jobs
[2]+  Running                  sleep 1000 &
```

관리 목적으로 사용자 로그아웃

다양한 이유로 다른 사용자를 로그아웃해야 할 수 있습니다. 몇 가지 가능한 시나리오는 사용자가 보안을 위반한 경우, 사용자가 리소스를 과도하게 사용한 경우, 사용자 시스템이 응답하지 않는 경우 또는 자료에 대한 사용자의 액세스 권한이 부적절한 경우입니다. 이러한 경우 관리 목적으로 신호를 사용하여 세션을 종료해야 합니다.

먼저 사용자를 로그아웃하기 위해 종료할 로그인 세션을 확인합니다. **w** 명령을 사용하여 사용자 로그인 및 현재 실행 중인 프로세스를 표시합니다. **TTY** 및 **FROM** 열을 통해 단을 세션을 확인합니다.

모든 사용자 로그인 세션은 터미널 장치(TTY)와 연결됩니다. 장치 이름이 **pts/N**인 경우 그래픽 터미널 창 또는 원격 로그인 세션과 연결된 의사 터미널입니다. **ttyN**인 경우에는 사용자가 시스템 콘솔, 대체 콘솔, 직접 연결된 다른 터미널 장치에 있습니다.

```
[user@host ~]$ w
12:43:06 up 27 min,  5 users,  load average: 0.03, 0.17, 0.66
USER     TTY      FROM          LOGIN@    IDLE   JCPU      PCPU WHAT
root     tty2          12:26   14:58   0.04s  0.04s -bash
bob      tty3          12:28   14:42   0.02s  0.02s -bash
user     pts/1  desktop.example.com 12:41   2.00s  0.03s  0.03s w
[user@host ~]$
```

세션 로그인 시간을 보고 시스템에 얼마나 오랫동안 연결되어 있었는지 확인합니다. 백그라운드 작업과 하위 프로세스를 포함하여 현재 작업에서 사용하는 CPU 리소스는 각 세션의 **JCPU** 열에 있습니다. 현재 프로그래드 프로세스 CPU 소모량은 **PCPU** 열에 표시됩니다.



참조

kill(1), **killall(1)**, **pgrep(1)**, **pkill(1)**, **pstree(1)**, **signal(7)** 및 **w(1)** 도움말 페이지

자세한 내용은

<https://www.gnu.org/software/libc/manual/pdf/libc.pdf#Signal%20Handling>
에서 Signal Handling을 참조하십시오.

자세한 내용은

<https://www.gnu.org/software/libc/manual/pdf/libc.pdf#Processes>
에서 Processes를 참조하십시오.

▶ 연습 가이드

프로세스 강제 종료

이 연습에서는 신호를 사용하여 프로세스를 관리하고 중지합니다.

결과

- 여러 개의 쉘 프로세스를 시작하고 중지합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start processes-kill
```

지침

- ▶ 1. **workstation** 시스템에서 두 개의 터미널 창을 나란히 엽니다. 이 섹션에서는 이러한 터미널을 왼쪽과 오른쪽이라고 합니다. 각 터미널에서 **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$
```

- ▶ 2. 왼쪽 터미널 쉘에서 **/home/student/bin** 디렉터리를 생성합니다. 새 디렉터리에 **instance** 쉘 스크립트를 생성합니다. 실행 가능하도록 스크립트 권한을 변경합니다.

- 2.1. **/home/student/bin** 디렉터리를 생성합니다.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. **/home/student/bin** 디렉터리에 **instance** 스크립트 파일을 생성합니다. **i** 키를 눌러 Vim 대화형 모드를 시작합니다. 파일에는 표시된 대로 다음 내용이 있어야 합니다. **:wq** 명령을 사용하여 파일을 저장합니다.

```
[student@servera ~]$ cd /home/student/bin
[student@servera bin]$ vim /home/student/bin/instance
#!/bin/bash
while true; do
    echo -n "$@" >> ~/instance_outfile
    sleep 5
done
```

**참고**

`instance` 스크립트는 프로세스가 종료될 때까지 실행됩니다. 명령줄 인수를 `~/instance_outfile` 파일에 5초마다 한 번 추가합니다.

- 2.3. `instance` 스크립트 파일을 실행 가능하게 만듭니다.

```
[student@servera ~]$ chmod +x /home/student/bin/instance
```

- ▶ 3. 왼쪽 터미널 쉘에서 `/home/student/bin/` 디렉터리로 변경합니다. `network`, `interface`, `connection` 인수를 전달하여 `instance` 스크립트 파일로 프로세스 3개를 시작합니다. 백그라운드에서 프로세스를 시작합니다.

```
[student@servera bin]$ instance network &
[1] 3460
[student@servera bin]$ instance interface &
[2] 3482
[student@servera bin]$ instance connection &
[3] 3516
```

- ▶ 4. 오른쪽 터미널 쉘에서 프로세스 3개가 모두 `/home/student/instance_outfile` 파일에 내용을 추가하고 있는지 확인합니다.

```
[student@servera ~]$ tail -f ~/instance_outfile
network interface network connection interface network connection interface
network
...output omitted...
```

- ▶ 5. 왼쪽 터미널 쉘에서 기존 작업을 표시합니다.

```
[student@servera bin]$ jobs
[1]  Running           instance network &
[2]- Running           instance interface &
[3]+ Running           instance connection &
```

- ▶ 6. 신호를 사용하여 `instance network` 프로세스를 일시 중단합니다. `instance network` 프로세스가 `Stopped`으로 설정되었는지 확인합니다. `network` 프로세스가 더 이상 `~/instance_outfile` 파일에 내용을 추가하지 않는지 확인합니다.

- 6.1. `instance network` 프로세스를 중지합니다. 프로세스가 중지되었는지 확인합니다.

```
[student@servera bin]$ kill -SIGSTOP %1
[1]+ Stopped           instance network
[student@servera bin]$ jobs
[1]+ Stopped           instance network
[2]  Running           instance interface &
[3]- Running           instance connection &
```

- 6.2. 오른쪽 터미널 쉘에서 `tail` 명령 출력을 봅니다. `network` 단어가 `~/instance_outfile` 파일에 더 이상 추가되지 않는지 확인합니다.

```
...output omitted...
interface connection interface connection interface connection interface
```

- ▶ 7. 왼쪽 터미널 쉘에서 `instance interface` 프로세스를 종료합니다. `instance interface` 프로세스가 사라졌는지 확인합니다. `instance interface` 프로세스 출력이 `~/instance_outfile` 파일에 더 이상 추가되지 않는지 확인합니다.

- 7.1. `instance interface` 프로세스를 종료합니다. 프로세스가 종료되었는지 확인합니다.

```
[student@servera bin]$ kill -SIGTERM %2
[student@servera bin]$ jobs
[1]+  Stopped                  instance network
[2]-  Terminated                instance interface
[3]-  Running                   instance connection &
```

- 7.2. 오른쪽 터미널 쉘에서 `tail` 명령 출력을 봅니다. `interface` 단어가 `~/instance_outfile` 파일에 더 이상 추가되지 않는지 확인합니다.

```
...output omitted...
connection connection connection connection connection connection connection
connection
```

- ▶ 8. 왼쪽 터미널 쉘에서 `instance network` 프로세스를 재개합니다. `instance network` 프로세스가 `Running`으로 설정되었는지 확인합니다. `instance network` 프로세스 출력이 `~/instance_outfile` 파일에 추가되었는지 확인합니다.

- 8.1. `instance network` 프로세스를 재개합니다. 프로세스가 `Running` 상태인지 확인합니다.

```
[student@servera bin]$ kill -SIGCONT %1
[student@servera bin]$ jobs
[1]+  Running                   instance network &
[3]-  Running                   instance connection &
```

- 8.2. 오른쪽 터미널 쉘에서 `tail` 명령 출력을 봅니다. `network` 단어가 `~/instance_outfile` 파일에 추가되었는지 확인합니다.

```
...output omitted...
network connection network connection network connection network connection
network connection
```

- ▶ 9. 왼쪽 터미널 쉘에서 나머지 두 작업을 종료합니다. 남은 작업이 없고 출력이 중지되었는지 확인합니다.

- 9.1. `instance network` 프로세스를 종료합니다. 다음에는 `instance connection` 프로세스를 종료합니다.

```
[student@servera bin]$ kill -SIGTERM %1
[student@servera bin]$ kill -SIGTERM %3
[1]+ Terminated instance network
[student@servera bin]$ jobs
[3]+ Terminated instance connection
```

- ▶ 10. 왼쪽 터미널 쉘에서, 열려 있는 모든 터미널 쉘에서 현재 실행 중인 프로세스를 표시합니다. **tail** 프로세스를 종료합니다. 프로세스가 더 이상 실행되지 않는지 확인합니다.

10.1. 현재 실행 중인 모든 프로세스를 표시합니다. **tail** 줄만 표시되도록 검색 범위를 좁힙니다.

```
[student@servera bin]$ ps -ef | grep tail
student 4581 31358 0 10:02 pts/0 00:00:00 tail -f instance_outfile
student 4869 2252 0 10:33 pts/1 00:00:00 grep --color=auto tail
```

10.2. **tail** 프로세스를 종료합니다. 프로세스가 더 이상 실행되지 않는지 확인합니다.

```
[student@servera bin]$ pkill -SIGTERM tail
[student@servera bin]$ ps -ef | grep tail
student 4874 2252 0 10:36 pts/1 00:00:00 grep --color=auto tail
```

10.3. 오른쪽 터미널 쉘에서 **tail** 명령이 더 이상 실행되지 않는지 확인합니다.

```
...output omitted...
network connection network connection network connection Terminated
[student@servera ~]$
```

- ▶ 11. 추가 터미널을 닫습니다. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera bin]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish processes-kill
```

이것으로 섹션을 완료합니다.

프로세스 활동 모니터링

목표

부하 평균을 정의하고 리소스를 많이 사용하는 서버 프로세스를 확인합니다.

부하 평균 설명

부하 평균은 일정 기간 동안 인식된 시스템 부하를 나타내기 위해 Linux 커널에서 제공하는 측정값입니다. 시스템 부하가 증가 또는 감소하는지를 확인하기 위해 보류 중인 시스템 리소스 요청 수를 대략적으로 측정하는데 사용할 수 있습니다.

커널은 실행 가능 및 무중단 상태의 프로세스 수를 기준으로 5초마다 현재 부하 수치를 수집합니다. 이 숫자는 누적되어 가장 최근의 1분, 5분, 15분 동안 지수 이동 평균으로 보고됩니다.

부하 평균 계산

부하 평균은 일정 기간 동안 인식된 시스템 부하를 나타냅니다. Linux는 CPU에서 실행할 준비가 된 프로세스 수와 디스크 또는 네트워크 I/O가 완료되기를 기다리고 있는 프로세스 수를 보고하여 부하 평균을 확인합니다.

- 부하 수치는 실행할 준비가 되고(프로세스 상태 **R**) 완료할 I/O에 대해 대기 중인(프로세스 상태 **D**) 프로세스의 평균 실행 수입니다.
- 일부 UNIX 시스템은 CPU 사용률 또는 실행 대기열 길이만 고려하여 시스템 부하를 표시합니다. Linux는 디스크 또는 네트워크 사용률도 포함합니다. 이러한 리소스의 사용량이 많으면 CPU 부하만큼 시스템 성능에 큰 영향을 미칠 수 있기 때문입니다. CPU 활동이 최소인 상태에서 부하 평균이 높은 경우 디스크 및 네트워크 활동을 조사합니다.

부하 평균은 다른 작업을 수행하기 전에 현재 요청이 완료되기를 기다리고 있는 프로세스 수를 대략적으로 측정한 값입니다. 요청은 CPU 시간에 대해 프로세스를 실행할 수 있습니다. 또는 완료할 중요한 디스크 I/O 작업에 대한 요청일 수 있으며, CPU가 유휴 상태인 경우에도 요청이 완료될 때까지 CPU에서 프로세스를 실행할 수 없습니다. 어느 쪽이든 시스템 부하가 영향을 받으며, 프로세스가 실행 대기 중이므로 시스템이 더 느리게 실행되는 것처럼 보입니다.

부하 평균 값 해석

uptime 명령은 현재 부하 평균을 표시하는 한 가지 방법입니다. 현재 시간, 시스템 가동 시간, 실행 중인 사용자 세션 수 및 현재 부하 평균을 출력합니다.

```
[user@host ~]$ uptime
15:29:03 up 14 min,  2 users,  load average: 2.92, 4.48, 5.20
```

부하 평균에 대한 세 개의 값은 마지막 1분, 5분, 15분에 대한 부하를 나타냅니다. 시스템 부하가 증가 또는 감소하는 것처럼 보이는지를 나타냅니다.

CPU를 기다리고 있는 프로세스가 부하 평균에 큰 영향을 주는 경우 대략적인 CPU당 부하 값을 계산하여 시스템에서 상당한 대기가 발생하는지 확인할 수 있습니다.

lscpu 명령을 사용하여 시스템에 있는 CPU 수를 확인합니다.

다음 예제에서 시스템은 코어당 두 개의 하이퍼스레드가 있는 듀얼 코어 단일 소켓 시스템입니다. Linux는 이 CPU 구성을 스케줄링 목적이 4개 CPU 시스템으로 취급합니다.

```
[user@host ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                4
On-line CPU(s) list:  0-3
Thread(s) per core:   2
Core(s) per socket:   2
Socket(s):             1
NUMA node(s):          1
...output omitted...
```

부하 수치에 대한 유일한 기여가 CPU 시간이 필요한 프로세스에서 발생한다고 가정해보십시오. 그런 다음 표시된 부하 평균 값을 시스템의 논리적 CPU 수로 나눌 수 있습니다. 값이 1 미만이면 적절한 리소스 사용량과 최소 대기 시간을 나타냅니다. 값이 1보다 크면 리소스 포화와 어느 정도의 처리 지연을 나타냅니다.

```
# From lscpu, the system has four logical CPUs, so divide by 4:
#                               load average: 2.92, 4.48, 5.20
#           divide by number of logical CPUs:    4    4    4
#                                         -----
#                               per-CPU load average: 0.73  1.12  1.30
#
# This system's load average appears to be decreasing.
# With a load average of 2.92 on four CPUs, all CPUs were in use ~73% of the time.
# During the last 5 minutes, the system was overloaded by ~12%.
# During the last 15 minutes, the system was overloaded by ~30%.
```

유휴 CPU 대기열의 부하 수치는 0입니다. CPU를 기다리는 각 프로세스가 부하 수치를 1씩 증가시킵니다. CPU에서 프로세스 1개가 실행되고 있으면 부하 수치는 1이고, 리소스(CPU)가 사용 중이지만 대기 중인 요청은 없습니다. 해당 프로세스가 1분 동안 실행되면 1분 부하 평균에 대한 기여는 1이 됩니다.

그러나 디스크 또는 네트워크 리소스 사용량이 많아 중요한 I/O를 위해 무중단 유휴 상태인 프로세스도 카운트에 포함되며 부하 평균을 높입니다. CPU 사용률을 나타내는 것은 아니지만 이러한 프로세스는 리소스를 기다리며, 리소스를 얻을 때까지 CPU에서 실행할 수 없기 때문에 대기열 카운트에 추가됩니다. 이 지표도 프로세스를 실행할 수 없게 하는 리소스 제한 때문에 시스템 부하로 간주됩니다.

리소스 포화 상태가 되기 전에는 대기열에서 대기 중인 작업이 거의 없으므로 부하 평균이 1 미만으로 유지됩니다. 리소스 포화 상태로 인해 요청이 대기열에 유지되고 부하 계산 루틴에서 카운트되는 경우에만 부하 평균이 증가합니다. 리소스 사용량이 100%에 근접하면 이후의 각 추가 요청은 서비스 대기 시간을 경험하기 시작합니다.

실시간 프로세스 모니터링

top 명령은 시스템 프로세스의 동적 뷰 및 요약 헤더와 프로세스 또는 스레드 목록을 차례로 표시합니다. 정적 **ps** 명령 출력과는 달리, **top** 명령은 구성 가능한 간격마다 계속해서 새로 고치고 열 순서 변경, 정렬, 강조 표시 기능을 제공합니다. **top** 설정을 영구적으로 변경할 수 있습니다. 기본 **top** 출력 열은 다음과 같습니다.

- 프로세스 ID(**PID**).
- 프로세스 소유자 사용자 이름(**USER**)

- 가상 메모리(**VIRT**)는 상주 세트, 공유 라이브러리, 매팅 또는 스와핑된 메모리 페이지를 포함하여 프로세스에서 사용하는 모든 메모리입니다(**ps** 명령에서 **VSZ** 레이블로 지정됨).
- 상주 메모리(**RES**)는 상주 공유 오브젝트를 포함하여 프로세스에서 사용하는 물리 메모리입니다(**ps** 명령에서 **RSS** 레이블로 지정됨).
- 프로세스 상태(**S**)는 다음 상태 중 하나일 수 있습니다.
 - **D** = 인터럽트 없는 유휴 상태
 - **R** = 실행 중 또는 실행 가능
 - **S** = 유휴 상태
 - **T** = 중지됨 또는 트레이스됨
 - **Z** = Zombie
- CPU 시간(**TIME**)은 프로세스 시작 이후의 총 처리 시간입니다. 모든 이전 하위 프로세스의 누적 시간을 포함하도록 토글할 수 있습니다.
- 프로세스 명령 이름(**COMMAND**).

top 명령의 기본 키 입력

키	목적
? or h	대화형 키 입력 도움말.
l, t, m	부하, 스레드 및 메모리 헤더 라인을 전환합니다.
1	헤더에 개별 CPU 또는 모든 CPU 요약을 표시하는 토글입니다.
s	새로 고침(화면) 속도를 소수 초 단위(예: 0.5, 1, 5)로 변경합니다.
b	Running 프로세스에 대한 반전 강조 표시를 전환합니다. 기본값은 굵은 글꼴입니다.
Shift+b	디스플레이, 헤더, 실행 중인 프로세스에 굵은 글꼴을 사용하도록 설정합니다.
Shift+h	스레드를 전환합니다. 프로세스 요약 또는 개별 스레드를 표시합니다.
u, Shift+u	사용자 이름으로 필터링합니다(유효, 실제).
Shift+m	메모리 사용량에 따라 프로세스 목록을 내림차순으로 정렬합니다.
Shift+p	프로세스 사용량에 따라 프로세스 목록을 내림차순으로 정렬합니다.
k	프로세스를 종료합니다. 메시지가 표시되면 PID , signal 를 차례로 입력합니다.
r	프로세스 우선 순위를 변경합니다. 메시지가 표시되면 PID , nice_value 를 차례로 입력합니다.
Shift+w	다음에 top 을 다시 시작할 때 사용할 현재 디스플레이 구성을 작성(저장)합니다.
q	종료
f	필드를 활성화 또는 비활성화하여 열을 관리합니다. top 의 정렬 필드를 설정할 수도 있습니다.



참고

보안 모드로 **top** 명령을 시작한 경우에는 **s**, **k**, **r** 키 입력을 사용할 수 없습니다.



참조

ps(1), **top(1)**, **uptime(1)** 및 **w(1)** 도움말 페이지

▶ 연습 가이드

프로세스 활동 모니터링

이 연습에서는 **top** 명령을 사용하여 실행 중인 프로세스를 검사하고 동적으로 제어합니다.

결과

- 실시간으로 프로세스를 관리합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start processes-monitor
```

지침

- ▶ 1. **workstation** 시스템에서 두 개의 터미널 창을 나란히 엽니다. 이 섹션에서는 이러한 터미널을 왼쪽과 오른쪽이라고 합니다. 각 터미널에서 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 왼쪽 터미널 쉘에서 **/home/student/bin** 디렉터리를 생성합니다. 새 디렉터리에 **monitor**라는 쉘 스크립트를 생성하여 인위적인 CPU 부하를 생성합니다. **monitor** 스크립트 파일을 실행 가능하게 만듭니다.

- 2.1. **/home/student/bin** 디렉터리를 생성합니다.

```
[student@servera ~]$ mkdir /home/student/bin
```

- 2.2. **/home/student/bin** 디렉터리에 스크립트 파일을 생성합니다.

```
[student@servera ~]$ vim /home/student/bin/monitor
#!/bin/bash
while true; do
    var=1
    while [[ var -lt 60000 ]]; do
        var=$($var+1)
    done
    sleep 1
done
```

**참고**

monitor 스크립트는 프로세스가 종료될 때까지 실행됩니다. 더하기 계산 6만 건을 수행하여 인위적인 CPU 부하를 생성합니다. CPU 부하를 생성한 후 1초 동안 유휴 상태였다가 변수를 재설정하고 반복합니다.

2.3. **monitor** 파일을 실행 가능하게 만듭니다.

```
[student@servera ~]$ chmod a+x /home/student/bin/monitor
```

▶ 3. 오른쪽 터미널 쉘에서 **top** 명령을 실행합니다. 창의 크기를 조정하여 명령 내용을 봅니다.

```
[student@servera ~]$ top
top - 12:13:03 up 11 days, 58 min, 3 users, load average: 0.00, 0.00, 0.00
Tasks: 113 total, 2 running, 111 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.0 sy, 0.0 ni, 99.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 1829.4 total, 1377.3 free, 193.9 used, 258.2 buff/cache
MiB Swap: 1024.0 total, 1024.0 free, 0.0 used. 1476.1 avail Mem

PID USER      PR  NI      VIRT      RES      SHR S %CPU %MEM     TIME+ COMMAND
5861 root      20   0          0      0      0 I  0.3    0.0  0:00.71 kworker/1:3-
events
6068 student   20   0  273564  4300  3688 R  0.3    0.2  0:00.01 top
  1 root      20   0 178680 13424  8924 S  0.0    0.7  0:04.03 systemd
  2 root      20   0          0      0      0 S  0.0    0.0  0:00.03 kthreadd
  3 root      20  -20          0      0      0 I  0.0    0.0  0:00.00 rcu_gp
...output omitted...
```

▶ 4. 왼쪽 터미널 쉘에서 이 가상 시스템의 논리 CPU 수를 확인합니다.

```
[student@servera ~]$ lscpu
Architecture:           x86_64
CPU op-mode(s):         32-bit, 64-bit
Byte Order:             Little Endian
CPU(s):                 2
...output omitted...
```

▶ 5. 왼쪽 터미널 쉘에서 **monitor** 스크립트 파일의 단일 인스턴스를 백그라운드에서 실행합니다.

```
[student@servera ~]$ monitor &
[1] 6071
```

▶ 6. 오른쪽 터미널 쉘에서 **top** 명령을 모니터링합니다. **l**, **t** 및 **m** 단일 키 입력을 사용하여 부하, 스레드 및 메모리 헤더 라인을 전환합니다. 이 동작을 관찰한 후 모든 헤더가 표시되는지 확인합니다.

▶ 7. **monitor** 프로세스의 PID(프로세스 ID)를 확인합니다. 프로세스의 CPU 백분율을 확인합니다. 25% 정도일 것으로 예상됩니다.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
071 student    20   0 222448  2964  2716 S 18.7  0.2  0:27.35 monitor
...output omitted...
```

부하 평균을 확인합니다. 현재 1분간의 부하 평균 값은 1 미만입니다. 관찰되는 값은 다른 가상 시스템 또는 가상 호스트에서 발생하는 리소스 경합의 영향을 받을 수 있습니다.

```
top - 12:23:45 up 11 days, 1:09, 3 users, load average: 0.21, 0.14, 0.05
```

- ▶ 8. 왼쪽 터미널 쉘에서 **monitor** 스크립트 파일의 두 번째 인스턴스를 백그라운드에서 실행합니다.

```
[student@servera ~]$ monitor &
[2] 6498
```

- ▶ 9. 오른쪽 터미널 쉘에서 두 번째 **monitor** 프로세스의 PID(프로세스 ID)를 확인합니다. 프로세스의 CPU 백분율을 봅니다. 역시 15~25% 정도일 것으로 예상됩니다.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
6071 student    20   0 222448  2964  2716 S 19.0  0.2  1:36.53 monitor
6498 student    20   0 222448  2996  2748 R 15.7  0.2  0:16.34 monitor
...output omitted...
```

1분간의 부하 평균을 다시 봅니다. 여전히 1 미만입니다. 새 워크로드에 맞게 계산이 조정되도록 1분 이상 기다립니다.

```
top - 12:27:39 up 11 days, 1:13, 3 users, load average: 0.36, 0.25, 0.11
```

- ▶ 10. 왼쪽 터미널 쉘에서 **monitor** 스크립트 파일의 세 번째 인스턴스를 백그라운드에서 실행합니다.

```
[student@servera ~]$ monitor &
[3] 6881
```

- ▶ 11. 오른쪽 터미널 쉘에서 세 번째 **monitor** 프로세스의 PID(프로세스 ID)를 확인합니다. 프로세스의 CPU 백분율을 봅니다. 다시 15~25% 정도일 것으로 예상됩니다.

```
[student@servera ~]$ top
PID USER      PR  NI    VIRT    RES    SHR S %CPU %MEM     TIME+ COMMAND
6881 student    20   0 222448  3032  2784 S 18.6  0.2  0:11.48 monitor
6498 student    20   0 222448  2996  2748 S 15.6  0.2  0:47.86 monitor
6071 student    20   0 222448  2964  2716 S 18.1  0.2  2:07.86 monitor
```

부하 평균을 1보다 크게 내보내려면 **monitor** 프로세스를 더 많이 시작해야 합니다. 강의실 설치에는 CPU 2개가 있으므로 프로세스 3개만으로는 스트레스를 주기에 충분하지 않습니다. 백그라운드에

6장 | 시스템 성능 튜닝

서 **monitor** 프로세스를 3개 더 시작합니다. 1분간의 부하 평균을 다시 봅니다. 이제 1보다 클 것으로 예상됩니다. 새 워크로드에 맞게 계산이 조정되도록 1분 이상 기다립니다.

```
[student@servera ~]$ monitor &
[4] 10708
[student@servera ~]$ monitor &
[5] 11122
[student@servera ~]$ monitor &
[6] 11338
```

```
top - 12:42:32 up 11 days,  1:28,  3 users,  load average: 1.23, 2.50, 1.54
```

▶ 12. 부하 평균 값 관찰을 마쳤으면 **top** 명령 내에서 각 **monitor** 프로세스를 종료합니다.

12.1. 오른쪽 터미널 쉘에서 **k** 키를 눌러 헤더 아래와 열 위의 프롬프트를 관찰합니다.

```
...output omitted...
PID to signal/kill [default pid = 11338]
```

12.2. 프롬프트는 목록 맨 위에 있는 **monitor** 프로세스를 선택합니다. **Enter** 키를 눌러 프로세스를 강제 종료합니다.

```
...output omitted...
Send pid 11338 signal [15/sigterm]
```

12.3. **Enter**를 다시 눌러 SIGTERM 신호 15를 확인합니다.

선택한 프로세스가 더 이상 **top** 명령에 없는지 확인합니다. PID가 있는 경우 이러한 단계를 반복하여 프로세스를 종료하고 메시지가 표시되면 SIGKILL 신호 9로 대체합니다.

6498	student	20	0	222448	2996	2748	R	22.9	0.2	5:31.47	monitor
6881	student	20	0	222448	3032	2784	R	21.3	0.2	4:54.47	monitor
11122	student	20	0	222448	2984	2736	R	15.3	0.2	2:32.48	monitor
6071	student	20	0	222448	2964	2716	S	15.0	0.2	6:50.90	monitor
10708	student	20	0	222448	3032	2784	S	14.6	0.2	2:53.46	monitor

▶ 13. 나머지 각 **monitor** 프로세스에 대해 이전 단계를 반복합니다. **top** 명령에 남은 **monitor** 프로세스가 없는지 확인합니다.

▶ 14. 오른쪽 터미널 쉘에서 **q** 키를 눌러 **top** 명령을 종료합니다. 오른쪽 터미널을 닫습니다.

▶ 15. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish processes-monitor
```

이것으로 섹션을 완료합니다.

튜닝 프로파일 조정

목표

tuned 데몬이 관리하는 튜닝 프로파일을 선택하여 시스템 성능을 최적화합니다.

시스템 튜닝

시스템 관리자는 다양한 사용 사례 워크로드에 따라 다양한 장치 설정을 조정하여 시스템 성능을 최적화 할 수 있습니다. **tuned** 데몬이 특정 워크로드 요구 사항을 반영하는 튜닝 프로파일을 사용하여 정적 및 동적으로 튜닝 조정을 적용합니다.

정적 튜닝 구성

tuned 데몬은 서비스가 시작될 때 또는 새 튜닝 프로파일을 선택할 때 시스템 설정을 적용합니다. 정적 튜닝은 **tuned** 데몬이 런타임에 적용하는 프로파일에서 미리 정의된 **kernel** 매개 변수를 구성합니다. 정적 튜닝에서는 **tuned** 데몬이 전반적인 성능 기대치에 맞게 커널 매개 변수를 설정하고, 활동 수준이 변경될 때 이러한 매개 변수를 조정하지 않습니다.

동적 튜닝 구성

동적 튜닝에서는 **tuned** 데몬이 시스템 활동을 모니터링하고, 런타임 동작 변경에 따라 설정을 조정합니다. 동적 튜닝은 선택한 튜닝 프로파일에 선언된 초기 설정으로 시작하여, 현재 워크로드에 맞게 지속적으로 튜닝을 조정합니다.

예를 들어 스토리지 장치는 시작 및 로그인 중에 높은 사용률을 보이지만 사용자 워크로드가 웹 브라우저 및 이메일 클라이언트 사용으로 구성된 경우에는 활동이 최소화됩니다. 마찬가지로, CPU 및 네트워크 장치는 업무일 중 사용량이 가장 많은 시간에 활동이 증가합니다. **tuned** 데몬은 이러한 구성 요소의 활동을 모니터링하고 매개 변수 설정을 조정하여 활동량이 많을 때는 성능을 극대화하고 활동량이 적을 때는 설정을 낮춥니다. 사전 정의된 튜닝 프로파일에서는 **tuned** 데몬이 사용하는 성능 매개 변수를 제공합니다.

tuned 데몬은 매개 변수 설정을 모니터링하고 조정하기 위해 각각 monitor 및 tuning 플러그인이라는 모듈을 사용합니다.

monitor 플러그인은 시스템을 분석하고 시스템에서 정보를 가져오며, tuning 플러그인은 이 정보를 동적 튜닝에 사용합니다. 현재 **tuned** 데몬에는 세 가지 monitor 플러그인이 포함되어 있습니다.

- **disk**: 모든 디스크 장치의 I/O 작업 수에 따라 디스크 부하를 모니터링합니다.
- **net**: 네트워크 카드당 전송되는 패킷 수에 따라 네트워크 부하를 모니터링합니다.
- **load**: 모든 CPU의 CPU 부하를 모니터링합니다.

tuning 플러그인은 개별 하위 시스템을 튜닝합니다. monitor 플러그인 데이터와 사전 정의된 튜닝 프로파일에서 제공하는 성능 매개 변수를 사용합니다. 특히 **tuned** 데몬은 다음 tuning 플러그인과 함께 제공됩니다.

- **disk**: 다양한 디스크 매개 변수(예: 디스크 스케줄러, 스픬 다운 시간 제한 또는 고급 전원 관리)를 설정합니다.
- **net**: 인터페이스 속도 및 WoL(Wake-on-LAN) 기능을 구성합니다.
- **cpu**: 다양한 CPU 매개 변수(예: CPU 관리자 또는 대기 시간)를 설정합니다.

동적 튜닝은 기본적으로 비활성화되어 있습니다. **/etc/tuned/tuned-main.conf** 구성 파일에서 **dynamic_tuning** 변수를 1로 설정하여 활성화할 수 있습니다. 동적 튜닝을 활성화하면 **tuned** 데몬이

6장 | 시스템 성능 튜닝

시스템을 주기적으로 모니터링하고 런타임 동작의 변화에 맞게 튜닝 설정을 조정합니다. `/etc/tuned/tuned-main.conf` 구성 파일에서 `update_interval` 변수를 사용하여 업데이트 빈도(초)를 설정할 수 있습니다.

```
[root@host ~]$ cat /etc/tuned/tuned-main.conf
...output omitted...
# Dynamically tune devices, if disabled only static tuning will be used.
dynamic_tuning = 1
...output omitted...
# Update interval for dynamic tunings (in seconds).
# It must be multiply of the sleep_interval.
update_interval = 10
...output omitted...
```

튜닝된 유틸리티

Red Hat Enterprise Linux 최소 설치에는 기본적으로 `tuned` 패키지가 포함됩니다. 다음 명령을 사용하여 패키지를 수동으로 설치하고 활성화할 수 있습니다.

```
[root@host ~]$ dnf install tuned
...output omitted...
[root@host ~]$ systemctl enable --now tuned
Created symlink /etc/systemd/system/multi-user.target.wants/tuned.service → /usr/
lib/systemd/system/tuned.service.
```

`tuned` 애플리케이션은 다음 범주의 프로파일을 제공합니다.

- Power-saving 프로필
- 성능 향상 프로파일

성능 향상 프로파일에는 다음 측면에 중점을 두는 프로파일이 포함됩니다.

- 스토리지 및 네트워크에 대한 낮은 대기 시간
- 스토리지 및 네트워크에 대한 높은 처리량
- 가상 시스템 성능
- 가상화 호스트 성능

다음 표에는 Red Hat Enterprise Linux 9과 함께 배포되는 튜닝 프로파일 목록이 있습니다.

Red Hat Enterprise Linux 9과 함께 배포되는 튜닝 프로파일

튜닝 프로파일	목적
<code>balanced</code>	절전과 성능 간에 절충이 필요한 시스템에 적합합니다.
<code>powersave</code>	최대 절전을 위해 시스템을 조정합니다.
<code>throughput-performance</code>	최대 처리량을 얻기 위해 시스템을 튜닝합니다.
<code>accelerator-performance</code>	<code>throughput-performance</code> 와 동일하게 튜닝하고 대기 시간도 100µs 미만으로 줄입니다.
<code>latency-performance</code>	전력 소비가 크더라도 대기 시간이 짧아야 하는 서버 시스템에 적합합니다.

튜닝 프로파일	목적
network-throughput	throughput-performance 프로파일에서 파생됩니다. 최대 네트워크 처리량을 얻기 위해 추가 네트워크 튜닝 매개 변수가 적용됩니다.
network-latency	latency-performance 프로파일에서 파생됩니다. 낮은 네트워크 대기 시간을 제공하기 위해 추가 네트워크 튜닝 매개 변수를 활성화합니다.
desktop	balanced 프로파일에서 파생됩니다. 대화형 애플리케이션의 빠른 응답을 제공합니다.
hpc-compute	latency-performance 프로파일에서 파생됩니다. 고성능 컴퓨팅에 이상적입니다.
virtual-guest	가상 시스템에서 실행할 경우 최대 성능을 얻기 위해 시스템을 튜닝합니다.
virtual-host	가상 시스템의 호스트로 사용될 경우 최대 성능을 얻기 위해 시스템을 튜닝합니다.
intel-sst	Intel Speed Select 기술 구성을 사용하는 시스템에 최적화되었습니다. 다른 프로파일의 오버레이로 사용합니다.
optimize-serial-console	직렬 콘솔의 응답성을 높입니다. 다른 프로파일의 오버레이로 사용합니다.

tuned 애플리케이션은 **/usr/lib/tuned** 및 **/etc/tuned** 디렉터리에 튜닝 프로파일을 저장합니다. 모든 프로파일에는 별도의 디렉터리가 있으며 디렉터리 내부에는 **tuned.conf** 기본 구성 파일과 기타 선택 파일이 있습니다.

```
[root@host ~]# cd /usr/lib/tuned
[root@host tuned]# ls
accelerator-performance  hpc-compute          network-throughput      throughput-
performance              balanced             intel-sst            optimize-serial-console virtual-
guest                  guest                latency-performance   powersave           virtual-
desktop                functions            network-latency       recommend.d
host                   host                powersave           virtual-
functions              tuned.conf
[root@host tuned]$ ls virtual-guest
tuned.conf
```

일반적인 **tuned.conf** 구성 파일은 다음과 같습니다.

```
[root@host tuned]# cat virtual-guest/tuned.conf
#
# tuned configuration
#
[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance
```

```
[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up. Workloads
# that mostly use file mappings may be able to use even higher values.
#
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 30

# Filesystem I/O is usually much more efficient than swapping, so try to keep
# swapping low. It's usually safe to go even lower than this on systems with
# server-grade storage.
vm.swappiness = 30
```

파일의 **[main]** 섹션에는 튜닝 프로파일 요약이 포함될 수 있습니다. 이 섹션에서는 **include** 매개 변수를 사용하여 프로파일이 참조 프로파일의 모든 설정을 상속하도록 할 수 있습니다.

이 구성 파일은 제공된 프로파일 중 하나를 기준으로 사용한 다음 구성할 매개 변수를 추가하거나 수정할 수 있으므로 새 튜닝 프로파일을 생성할 때 매우 유용합니다. 튜닝 프로파일을 생성하거나 수정하려면 **/usr/lib/tuned** 디렉터리에서 **/etc/tuned** 디렉터리로 튜닝 프로파일 파일을 복사한 다음 수정합니다. **/usr/lib/tuned** 및 **/etc/tuned** 디렉터리에 동일한 이름의 프로파일 디렉터리가 있는 경우 항상 후자가 우선합니다. 따라서 **/usr/lib/tuned** 시스템 디렉터리에 있는 파일을 직접 수정하지 마십시오.

tuned.conf 파일의 나머지 섹션에서는 tuning 플러그인을 사용하여 시스템의 매개 변수를 수정합니다. 위 예제에서 **[sysctl]** 섹션은 **sysctl** 플러그인을 통해 **vm.dirty_ratio** 및 **vm.swappiness** 커널 매개 변수를 수정합니다.

명령줄에서 프로파일 관리

tuned-adm 명령은 **tuned** 데몬의 설정을 변경하는 데 사용됩니다. **tuned-adm** 명령은 현재 설정을 쿼리하거나, 사용 가능한 프로파일을 나열하거나, 시스템에 맞는 튜닝 프로파일을 권장하거나, 프로파일을 직접 변경하거나, 튜닝을 해제합니다.

tuned-adm active 명령을 사용하여 현재 활성 상태인 튜닝 프로파일을 확인할 수 있습니다.

```
[root@host ~]# tuned-adm active
Current active profile: virtual-guest
```

tuned-adm list 명령은 기본 제공 프로파일과 사용자가 생성한 튜닝 프로파일을 둘 다 포함하여 사용 가능한 모든 튜닝 프로파일을 나열합니다.

```
[root@host ~]# tuned-adm list
Available profiles:
- accelerator-performance      - Throughput performance based tuning with ...
- balanced                   - General non-specialized tuned profile
- desktop                    - Optimize for the desktop use-case
- hpc-compute                - Optimize for HPC compute workloads
- intel-sst                  - Configure for Intel Speed Select Base Frequency
- latency-performance        - Optimize for deterministic performance at ...
```

```
- network-latency           - Optimize for deterministic performance at ...
...output omitted...
Current active profile: virtual-guest
```

지정된 프로필에 대한 정보를 보려면 **tuned-adm profile_info** 명령을 사용합니다.

```
[root@host ~]$ tuned-adm profile_info network-latency
Profile name:
network-latency

Profile summary:
Optimize for deterministic performance at the cost of increased power consumption,
focused on low latency network performance
...output omitted..
```

프로필이 지정되지 않은 경우 **tuned-adm profile_info** 명령은 활성 튜닝 프로필에 대한 정보를 표시합니다.

```
[root@host ~]$ tuned-adm active
Current active profile: virtual-guest
[root@host ~]$ tuned-adm profile_info
Profile name:
virtual-guest

Profile summary:
Optimize for running inside a virtual guest
...output omitted..
```

시스템의 현재 튜닝 요구 사항에 더 잘 맞는 다른 활성 프로파일로 전환하려면 **tuned-adm profile filename** 명령을 사용합니다.

```
[root@host ~]$ tuned-adm profile throughput-performance
[root@host ~]$ tuned-adm active
Current active profile: throughput-performance
```

tuned-adm recommend 명령은 시스템에 맞는 튜닝 프로파일을 권장할 수 있습니다. 이 메커니즘은 설치 후 기본 프로파일을 결정하는 데 사용됩니다.

```
[root@host ~]$ tuned-adm recommend
virtual-guest
```



참고

tuned-adm recommend 명령은 시스템이 가상 시스템인지의 여부 및 시스템 설치 중 선택한 기타 사전 정의된 범주를 포함하여 다양한 시스템 특성에 대한 권장 사항을 기반으로 합니다.

현재 프로파일에서 적용한 설정 변경 사항을 되돌리려면 다른 프로파일로 전환하거나 tuned 데몬을 비활성화합니다. **tuned-adm off** 명령을 사용하여 tuned 애플리케이션 튜닝 활동을 끕니다.

```
[root@host ~]$ tuned-adm off
[root@host ~]$ tuned-adm active
No current active profile.
```

웹 콘솔을 사용하여 프로파일 관리

웹 콘솔을 사용하여 시스템 성능 프로파일을 관리하려면 로그인하여 권한을 에스컬레이션해야 합니다. 권한 에스컬레이션 모드에서는 사용자가 관리 권한으로 시스템 성능 프로파일을 수정하는 명령을 실행할 수 있습니다. 튜닝 프로필을 변경하면 일부 시스템 매개 변수가 수정되므로 관리 권한으로 수행해야 합니다.

웹 콘솔에서 **Limited access** 또는 **Turn on administrative access** 버튼을 클릭하여 관리 액세스 모드로 전환할 수 있습니다. 그런 다음 메시지가 표시되면 암호를 입력합니다. 권한을 에스컬레이션하면 **Limited access** 버튼이 **Administrative access**로 변경됩니다. 보안을 위해 시스템에서 관리 권한이 필요한 작업을 완료한 후에는 항상 제한된 액세스 모드로 다시 전환해야 합니다.

권한 있는 사용자로서 왼쪽 탐색 모음에서 **Overview** 메뉴 옵션을 클릭합니다. **Performance profile** 필드에 현재 활성화된 프로파일이 표시됩니다.

The screenshot shows the Red Hat Web Console interface. At the top, there are navigation links: 'Administrative access' (highlighted in red), 'Help', and 'Session'. Below the header, there are four main sections: 'Health', 'Usage', 'System information', and 'Configuration'. The 'Configuration' section is expanded, showing various system settings. Under 'Performance profile', the value 'balanced' is displayed and highlighted with a red box. Other settings in this section include 'Hostname' (workstation.lab.example.com), 'System time' (May 19, 2022, 1:56 PM), 'Domain' (Join domain), and 'Secure shell keys' (Show fingerprints).

그림 6.1: 활성 성능 프로파일

다른 프로파일을 선택하려면 활성 프로파일 링크를 클릭합니다. **Change performance profile** 사용자 인터페이스에서 프로파일 목록을 스크롤하여 시스템 목적에 가장 적합한 프로파일을 선택하고 **Change profile** 버튼을 클릭합니다.

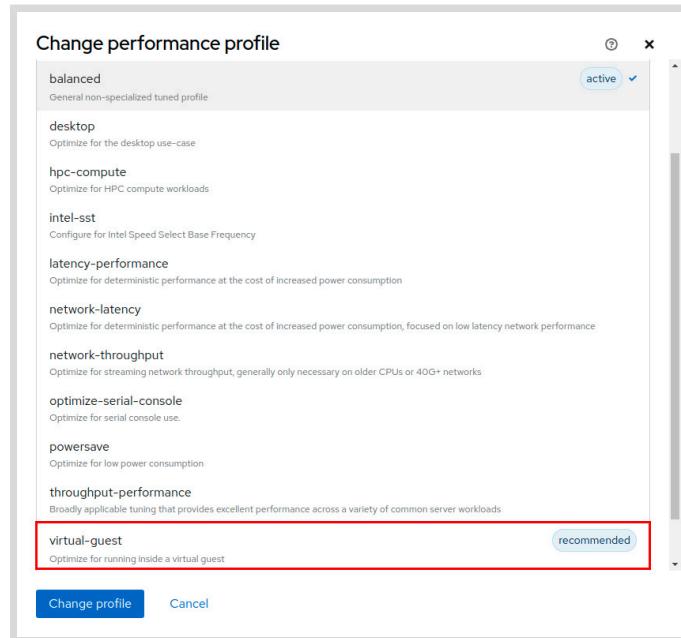


그림 6.2: 기본 성능 프로파일 선택

변경 사항을 확인하려면 주 **Overview** 페이지로 돌아간 다음, **Performance profile** 필드에 활성 프로파일이 표시되는지 확인합니다.



참조

`tuned(8)`, `tuned.conf(5)`, `tuned-main.conf(5)`, `tuned-adm(1)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/monitoring_and_managing_system_status_and_performance/index에 있는 Monitoring and Managing System Status and Performance 가이드를 참조하십시오.

▶ 연습 가이드

튜닝 프로파일 조정

이 연습에서는 **tuned** 서비스를 활성화하고 튜닝 프로파일을 적용하여 서버 성능을 튜닝합니다.

결과

- 튜닝 프로파일을 사용하도록 시스템을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start tuning-profiles
```

지침

- ▶ 1. **student** 사용자로 **servera**에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **tuned** 패키지가 설치되고 활성화 및 시작되었는지 확인합니다.

2.1. **tuned** 패키지가 설치되었는지 확인합니다.

```
[student@servera ~]$ dnf list tuned
...output omitted...
Installed Packages
tuned.noarch           2.18.0-1.el9          @System
```

2.2. 서비스가 활성화되었는지 확인합니다.

```
[student@servera ~]$ systemctl is-enabled tuned
enabled
```

2.3. 서비스가 현재 실행 중인지 확인합니다.

```
[student@servera ~]$ systemctl is-active tuned
active
```

- ▶ 3. 사용 가능한 튜닝 프로파일을 나열하고 활성 프로파일을 식별합니다.

```
[student@servera ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- accelerator-performance      - Throughput performance based tuning with disabled
      higher latency STOP states
- balanced                    - General non-specialized tuned profile
- desktop                     - Optimize for the desktop use-case
- hpc-compute                 - Optimize for HPC compute workloads
- intel-sst                   - Configure for Intel Speed Select Base Frequency
- latency-performance         - Optimize for deterministic performance at the cost
      of increased power consumption
- network-latency             - Optimize for deterministic performance at the cost
      of increased power consumption, focused on low latency network performance
- network-throughput          - Optimize for streaming network throughput,
      generally only necessary on older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                   - Optimize for low power consumption
- throughput-performance      - Broadly applicable tuning that provides excellent
      performance across a variety of common server workloads
- virtual-guest               - Optimize for running inside a virtual guest
- virtual-host                - Optimize for running KVM guests
Current active profile: virtual-guest
```

- ▶ 4. 현재 활성 프로파일 **virtual-guest**의 **tuned.conf** 구성 파일을 검토합니다. **/usr/lib/tuned/virtual-guest** 디렉터리에서 **tuned.conf** 구성 파일을 찾을 수 있습니다. **virtual-guest** 튜닝 프로파일은 **throughput-performance** 프로파일을 기반으로 하지 만 **vm.dirty_ratio** 및 **vm.swappiness** 매개 변수에 대해 다른 값을 설정합니다. **virtual-guest** 튜닝 프로파일이 이러한 값을 해당 시스템에 적용하는지 확인합니다.

- 4.1. **/usr/lib/tuned/virtual-guest** 디렉터리에서 **virtual-guest** 구성 파일을 검토합니다. **vm.dirty_ratio** 및 **vm.swappiness** 매개 변수의 값을 확인합니다.

```
[student@servera ~]$ cat /usr/lib/tuned/virtual-guest/tuned.conf
#
# tuned configuration
#
[main]
summary=Optimize for running inside a virtual guest
include=throughput-performance

[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up. Workloads
# that mostly use file mappings may be able to use even higher values.
#
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 30

# Filesystem I/O is usually much more efficient than swapping, so try to keep
```

```
# swapping low. It's usually safe to go even lower than this on systems with
# server-grade storage.
vm.swappiness = 30
```

4.2. 튜닝 프로파일이 이러한 값을 해당 시스템에 적용하는지 확인합니다.

```
[student@servera ~]$ sysctl vm.dirty_ratio
vm.dirty_ratio = 30
[student@servera ~]$ sysctl vm.swappiness
vm.swappiness = 30
```

- ▶ 5. `virtual-guest` 상위의 튜닝 프로파일 `throughput-performance`의 `tuned.conf` 구성 파일을 검토합니다. `/usr/lib/tuned/throughput-performance` 디렉터리에서 찾을 수 있습니다. `throughput-performance` 튜닝 프로파일은 `vm.dirty_ratio` 및 `vm.swappiness` 매개 변수에 다른 값을 설정하지만 `virtual-guest` 프로파일이 해당 값을 덮어씁니다. `virtual-guest` 튜닝 프로파일이 `vm.dirty_background_ratio` 매개 변수에 `throughput-performance` 프로파일에서 상속한 값을 적용하는지 확인합니다.

- 5.1. `/usr/lib/tuned/throughput-performance` 디렉터리에서 `throughput-performance` 구성 파일을 검토합니다. `vm.dirty_ratio`, `vm.swappiness` 및 `vm.dirty_background_ratio` 매개 변수의 값을 확인합니다.

```
[student@servera ~]$ cat /usr/lib/tuned/throughput-performance/tuned.conf
#
# tuned configuration
#

[main]
summary=Broadly applicable tuning that provides excellent performance across a
variety of common server workloads

...output omitted...

[sysctl]
# If a workload mostly uses anonymous memory and it hits this limit, the entire
# working set is buffered for I/O, and any more write buffering would require
# swapping, so it's time to throttle writes until I/O can catch up. Workloads
# that mostly use file mappings may be able to use even higher values.
#
# The generator of dirty data starts writeback at this percentage (system default
# is 20%)
vm.dirty_ratio = 40

# Start background writeback (via writeback threads) at this percentage (system
# default is 10%)
vm.dirty_background_ratio = 10

# PID allocation wrap value. When the kernel's next PID value
# reaches this value, it wraps back to a minimum PID value.
# PIDs of value pid_max or larger are not allocated.
#
# A suggested value for pid_max is 1024 * <# of cpu cores/threads in system>
# e.g., a box with 32 cpus, the default of 32768 is reasonable, for 64 cpus,
```

```
# 65536, for 4096 cpus, 4194304 (which is the upper limit possible).
#kernel.pid_max = 65536

# The swappiness parameter controls the tendency of the kernel to move
# processes out of physical memory and onto the swap disk.
# 0 tells the kernel to avoid swapping processes out of physical memory
# for as long as possible
# 100 tells the kernel to aggressively swap processes out of physical memory
# and move them to swap cache
vm.swappiness=10

...output omitted...
```

- 5.2. **virtual-guest** 튜닝 프로파일이 상속한 **vm.dirty_background_ratio** 매개 변수를 적용하는지 확인합니다.

```
[student@servera ~]$ sysctl vm.dirty_background_ratio
vm.dirty_background_ratio = 10
```

- ▶ 6. 현재 활성 튜닝 프로파일을 **throughput-performance**로 변경하고 결과를 확인합니다. **vm.dirty_ratio** 및 **vm.swappiness** 매개 변수가 **throughput-performance** 구성 파일의 값으로 변경되는지 확인합니다.

- 6.1. 현재 활성 튜닝 프로파일을 변경합니다.

```
[student@servera ~]$ sudo tuned-adm profile throughput-performance
```

- 6.2. **throughput-performance**가 활성 튜닝 프로파일인지 확인합니다.

```
[student@servera ~]$ sudo tuned-adm active
Current active profile: throughput-performance
```

- 6.3. **vm.dirty_ratio** 및 **vm.swappiness** 매개 변수의 값을 확인합니다.

```
[student@servera ~]$ sysctl vm.dirty_ratio
vm.dirty_ratio = 40
[student@servera ~]$ sysctl vm.swappiness
vm.swappiness = 10
```

- ▶ 7. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish tuning-profiles
```

이것으로 섹션을 완료합니다.

프로세스 스케줄링에 미치는 영향

목표

nice 및 **renice** 명령을 사용하여 특정 프로세스에 우선순위를 지정하거나 해제합니다.

Linux 프로세스 스케줄링

최신 컴퓨터 시스템은 여러 명령 스레드를 동시에 실행할 수 있는 멀티 코어, 멀티 스레드 CPU를 사용합니다. 가장 큰 고성능 슈퍼 컴퓨터는 CPU당 처리 코어 및 스레드 구조가 수백 개인 CPU를 수백 또는 수천 개 포함 할 수 있으며, 수백만 개의 명령 스레드를 병렬로 처리할 수 있습니다. 단일 사용자가 여러 애플리케이션을 실행하는 경우 일반적인 데스크탑 시스템이나 개인용 워크스테이션은 CPU 활동으로 포화 상태가 될 수 있지만, 정확한 규모로 적절하게 구성된 워크스테이션은 사용자가 의도한 워크로드에 맞게 설계됩니다. 그러나 일반적인 엔터프라이즈 또는 인터넷 서버는 사용자 및 애플리케이션의 요청을 초당 수백 또는 수천 개씩 처리 하므로 CPU가 쉽게 상태가 될 수 있습니다. CPU 부하가 있는 모든 시스템은 스레드를 즉시 예약하기 위해 사용 가능한 시스템 CPU 처리 장치보다 더 많은 프로세스 스레드를 처리해야 하는 시나리오를 경험합니다.

Linux 및 기타 운영 체제는 타임 슬라이싱 또는 멀티태스킹이라는 기술을 사용하여 프로세스를 관리합니다. 운영 체제의 프로세스 스케줄러는 사용 가능한 각 CPU 코어의 프로세스 스레드를 빠르게 전환합니다. 이 동작은 상당한 수의 프로세스가 동시에 실행되고 있다는 인상을 줍니다.

프로세스 우선순위

프로세스 우선순위는 각 프로세스의 중요도를 설정합니다. Linux에서는 CPU 처리 시간을 확보하기 위해 프로세스 구성 및 우선순위 지정 규칙을 정의하는 스케줄링 정책을 구현합니다. 다양한 Linux 스케줄링 정책이 대화형 애플리케이션 요청, 비 대화형 배치 애플리케이션 처리, 실시간 애플리케이션 요구 사항을 처리하도록 설계되었습니다. 실시간 스케줄링 정책은 여전히 프로세스 우선순위와 대기열을 사용합니다. 하지만 최근의 비 실시간(일반) 스케줄링 정책에서는 CPU 시간을 기다리는 프로세스를 바이너리 검색 트리로 구성하는 CFS(완전 공정 스케줄러)를 사용합니다. 이 프로세스 우선순위는 **SCHED_NORMAL** 또는 **SCHED_OTHER** 정책을 기본 스케줄링 정책으로 사용합니다.

모든 시스템 실시간 프로세스에 일반 프로세스보다 높은 우선순위가 지정되도록 **SCHED_NORMAL** 정책하에 실행 중인 프로세스에는 정적 실시간 우선순위 0이 할당됩니다. 그러나 이 정적 우선순위 값은 CPU 스케줄링을 위해 일반 프로세스 스레드를 구성할 때 포함되지 않습니다. 대신 CFS 스케줄링 알고리즘에서 일반 프로세스 스레드를 이전에 사용한 CPU 시간이 가장 짧은 첫 번째 항목부터 누적 CPU 시간이 가장 긴 마지막 항목 까지 시간 가중 바이너리 트리로 정렬합니다.

nice 값

바이너리 트리의 순서는 사용자가 수정할 수 있는 프로세스별 nice 값에 따라 추가로 영향을 받습니다. 해당 값은 범위가 -20(우선순위 증가)에서 19(우선순위 감소)까지이고 기본값은 0입니다. 프로세스는 해당 상위 프로세스에서 시작 nice 값을 상속합니다. 모든 사용자는 nice 값을 조정하여 우선순위를 낮출 수 있지만 **root** 만이 값을 사용하여 우선순위를 높일 수 있습니다.

nice 값이 클수록 프로세스 우선순위가 기본값에서 감소했음을 나타내거나 해당 프로세스가 다른 프로세스 보다 더 적합하게 되었을 수 있습니다. nice 값이 작을수록 프로세스 우선순위가 기본값에서 증가했음을 나타내거나 해당 프로세스가 다른 프로세스보다 덜 적합하게 되었을 수 있습니다.

nice 값을 늘리면 스레드의 위치가 낮아지고, 값을 줄이면 스레드의 위치가 높아집니다.

**중요**

일반적으로 우선순위는 프로세스 스레드가 수신하는 CPU 시간을 간접적으로만 결정합니다. 사용 가능한 CPU 용량이 있어 포화되지 않은 시스템의 모든 프로세스는 CPU 시간에 각 프로세스에 필요한 만큼 즉시 예약됩니다. 바이너리 트리에서 관리하는 상대적 프로세스 중요도는 어떤 스레드가 선택되어 CPU에 먼저 배치되는지만 결정합니다.

CPU 처리 장치보다 대기 중인 스레드가 더 많은 CPU 포화 시스템에서는 모든 CPU 장치가 사용될 때까지 우선순위가 높은(nice 값이 작음) 프로세스 스레드가 먼저 배치되고, 우선순위가 가장 낮은(nice 값이 큼) 스레드는 먼저 바이너리 트리에서 대기해야 합니다. 그러나 완전 공정 스케줄러는 프로세스 중요도, nice 값, 이전의 누적 CPU 시간이 균형을 이루도록 설계되었으며 모든 프로세스에서 공정한 CPU 시간을 확보하도록 바이너리 트리를 동적으로 조정합니다.

nice 값 수정 권한

권한이 있는 사용자는 프로세스의 nice 값을 줄여 프로세스를 덜 적합하게 만들 수 있습니다. 그런 다음 프로세스는 이진 트리의 상위에 반복적으로 배치되므로 더 자주 예약됩니다. 포화된 시스템에서는 다른 프로세스에서 사용할 수 있는 전체 CPU 시간이 줄어듭니다.

권한이 없는 사용자는 자신의 프로세스에서 nice 값을 늘릴 수만 있습니다. 그러면 자신의 프로세스 적합도가 높아져 프로세스가 바이너리 트리에서 더 낮은 위치에 배치됩니다. 권한이 없는 사용자는 중요도를 높이기 위해 프로세스의 nice 값을 줄일 수 없으며 다른 사용자의 프로세스에 해당하는 nice 값을 조정할 수도 없습니다.

nice 값 보기

nice 값은 우선순위 값에 매핑되며 이 두 값은 프로세스 목록 명령에서 볼 수 있습니다. nice 값 -20은 **top** 명령에서 우선순위 0에 매핑됩니다. nice 값 19는 **top** 명령에서 우선순위 39에 매핑됩니다.

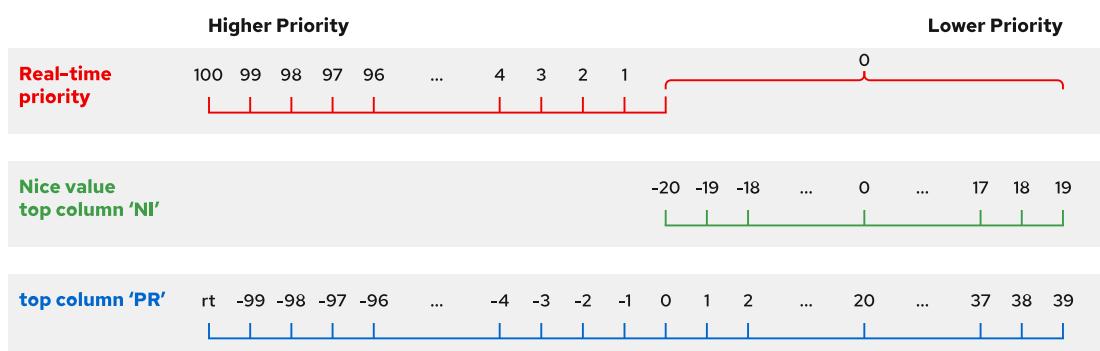


그림 6.3: **top** 명령에서 보고하는 우선순위 및 nice 값

앞의 그림에서 nice 값은 **top** 명령에서 사용하는 우선순위 값에 맞게 정렬됩니다. **top** 명령은 PR 열에 프로세스 우선순위를 표시하고 NI 열에 nice 값을 표시합니다. 실시간 프로세스 우선순위를 음수로 표시하는 **top** 우선순위 번호 지정 체계는 내부 우선순위 알고리즘의 레거시입니다.

다음 출력은 **top** 명령의 요약과 일부 프로세스 목록입니다.

```
Tasks: 192 total, 1 running, 191 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.0 us, 1.6 sy, 0.0 ni, 96.9 id, 0.0 wa, 0.0 hi, 1.6 si, 0.0 st
MiB Mem : 5668.6 total, 4655.6 free, 470.1 used, 542.9 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 4942.6 avail Mem

 PID USER      PR  NI      VIRT      RES     SHR S %CPU %MEM     TIME+ COMMAND

```

1 root	20	0	172180	16232	10328	S	0.0	0.3	0:01.49	systemd
2 root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3 root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_gp
4 root	0	-20	0	0	0	I	0.0	0.0	0:00.00	rcu_par_gp

ps 명령은 기본 형식 지정 옵션을 사용할 때 프로세스 nice 값을 표시합니다.

다음 **ps** 명령은 모든 프로세스를 프로세스 ID, 프로세스 이름, nice 값, 스케줄링 클래스와 함께 나열합니다. 프로세스는 nice 값을 기준으로 내림차순으로 정렬됩니다. **CLS** 스케줄링 클래스 열에서 **TS**는 시간 공유를 나타내며, **SCHED_NORMAL**을 포함한 일반 스케줄링 정책의 또 다른 이름입니다. 기타 **CLS** 값(예: 선입 선출의 경우 **FF**, 라운드 로빈의 경우 **RR**)은 실시간 프로세스를 나타냅니다. 실시간 프로세스에는 **NI** 열에 대시(-)로 표시된 nice 값이 할당되지 않습니다. 고급 스케줄링 정책은 Red Hat Performance Tuning: Linux in Physical, Virtual, and Cloud (RH442) 과정에서 설명합니다.

```
[user@host ~]$ ps axo pid,comm,nice,cls --sort=-nice
  PID COMMAND          NI  CLS
    33 khugepaged      19   TS
    32 ksmd            5   TS
  814 rtkit-daemon    1   TS
    1 systemd           0   TS
    2 kthreadd          0   TS
    5 kworker/0:0:cgr   0   TS
    7 kworker/0:1:rcu   0   TS
    8 kworker/u4:0:ev   0   TS
   15 migration/0      -   FF
...output omitted...
```

사용자가 설정한 nice 값으로 프로세스 시작

프로세스가 생성되면 상위 nice 값을 상속받습니다. 명령줄에서 프로세스를 시작하는 경우 해당 쉘 프로세스의 nice 값을 상속받습니다. 일반적으로 새 프로세스는 기본 nice 값 0으로 실행됩니다.

다음 예제에서는 쉘에서 프로세스를 시작하고 프로세스의 nice 값을 표시합니다. 요청된 출력을 지정하려면 **ps** 명령에 PID 옵션을 사용합니다.



참고

이 명령은 리소스 소비가 적기 때문에 데모용으로 선택되었습니다.

```
[user@host ~]$ sleep 60 &
[1] 2667
[user@host ~]$ ps -o pid,comm,nice 2667
  PID COMMAND          NI
  2667 sleep            0
```

모든 사용자가 **nice** 명령을 사용하여 기본 nice 값 또는 더 큰 nice 값을 사용하여 명령을 시작할 수 있습니다. 기본적으로 더 큰 값을 설정하면 새 프로세스의 우선순위가 현재 작업 중인 쉘보다 낮고 현재 대화형 세션에 영향을 미칠 가능성이 작아집니다.

다음 예제에서는 기본 nice 값을 사용하여 동일한 명령을 백그라운드 작업으로 시작하고 프로세스의 nice 값을 표시합니다.

```
[user@host ~]$ nice sleep 60 &
[1] 2736
[user@host ~]$ ps -o pid,comm,nice 2736
 PID COMMAND      NI
 2736 sleep      10
```

nice 명령 **-n** 옵션을 사용하여 시작 프로세스에 사용자 정의 nice 값을 적용합니다. 기본값은 프로세스의 현재 nice 값에 10을 더한 것입니다. 다음 예제에서는 사용자 정의 nice 값 **15**를 사용하여 백그라운드 작업을 시작하고 그 결과를 표시합니다.

```
[user@host ~]$ nice -n 15 sleep 60 &
[1] 2673
[user@host ~]$ ps -o pid,comm,nice 2740
 PID COMMAND      NI
 2740 sleep      15
```

기존 프로세스의 nice 값 변경

renice 명령을 사용하여 기존 프로세스의 nice 값을 변경할 수 있습니다. 이 예제에서는 위 예제의 프로세스 ID를 사용하여 현재 nice 값 15를 새 nice 값 19로 변경합니다.

```
[user@host ~]$ renice -n 19 2740
2740 (process ID) old priority 15, new priority 19
```

또한 **top** 명령을 사용하여 기존 프로세스의 nice 값을 변경할 수 있습니다. **top** 대화형 인터페이스에서 **r** 키를 눌러 **renice** 명령에 액세스합니다. 프로세스 ID를 입력한 다음 새 nice 값을 입력합니다.



참조

[nice\(1\)](#), [renice\(1\)](#), [top\(1\)](#), [sched_setscheduler\(2\)](#) 도움말 페이지

▶ 연습 가이드

프로세스 스케줄링에 미치는 영향

이 연습에서는 **nice** 및 **renice** 명령을 사용하여 프로세스의 스케줄링 우선순위를 조정하고 프로세스 실행에 미치는 영향을 확인합니다.

결과

- 프로세스의 스케줄링 우선순위를 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start tuning-procscheduling
```



중요

이 연습에서는 의도적으로 상당한 CPU 리소스를 사용하면서 장치 파일에 무한 체크 셈을 수행하는 명령을 사용합니다.

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **servera** 시스템의 CPU 코어 수를 확인한 다음 각 코어에 대해 **sha1sum /dev/zero &** 명령 인스턴스 두 개를 시작합니다.

- 2.1. **grep** 명령을 사용하여 **/proc/cpuinfo** 파일에서 기존 가상 프로세서(CPU 코어) 수를 구문 분석합니다.

```
[student@servera ~]$ grep -c '^processor' /proc/cpuinfo
2
```

- 2.2. 루프 명령을 사용하여 **sha1sum /dev/zero &** 명령 인스턴스를 여러 개 시작합니다. 위 단계에 표시된 각 가상 프로세서에 대해 인스턴스 두 개를 시작합니다. 이 예제에서 **for** 반복문은 인스턴스 네 개를 생성합니다. 출력에 표시되는 PID 값은 예제와 다를 수 있습니다.

```
[student@servera ~]$ for i in {1..4}; do sha1sum /dev/zero & done
[1] 1132
[2] 1133
[3] 1134
[4] 1135
```

- ▶ 3. 각 **sha1sum** 프로세스에 대해 백그라운드 작업이 실행되고 있는지 확인합니다.

```
[student@servera ~]$ jobs
[1]  Running          sha1sum /dev/zero &
[2]  Running          sha1sum /dev/zero &
[3]- Running          sha1sum /dev/zero &
[4]+ Running          sha1sum /dev/zero &
```

- ▶ 4. **ps** 및 **pgrep** 명령을 사용하여 각 **sha1sum** 프로세스의 CPU 사용률을 표시합니다.

```
[student@servera ~]$ ps u $(pgrep sha1sum)
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
student   1132 49.6  0.1 225336  2288 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1133 49.6  0.1 225336  2296 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1134 49.6  0.1 225336  2264 pts/0    R    11:40  2:40 sha1sum /dev/zero
student   1135 49.6  0.1 225336  2280 pts/0    R    11:40  2:40 sha1sum /dev/zero
```

- ▶ 5. **sha1sum** 프로세스를 모두 종료한 다음 실행 중인 작업이 없는지 확인합니다.

- 5.1. **pkill** 명령을 사용하여 이름 패턴이 **sha1sum** 인 실행 중인 프로세스를 모두 종료합니다.

```
[student@servera ~]$ pkill sha1sum
[2]  Terminated      sha1sum /dev/zero
[4]+  Terminated      sha1sum /dev/zero
[1]-  Terminated      sha1sum /dev/zero
[3]+  Terminated      sha1sum /dev/zero
```

- 5.2. 실행 중인 작업이 없는지 확인합니다.

```
[student@servera ~]$ jobs
[student@servera ~]$
```

- ▶ 6. **sha1sum /dev/zero &** 명령의 인스턴스를 여러 개 시작한 다음 nice 수준이 10인 **sha1sum /dev/zero &** 인스턴스를 하나 더 시작합니다. 최소한 시스템의 가상 프로세서 수만큼 인스턴스를 시작하십시오. 이 예제에서는 일반 인스턴스 세 개와 nice 수준이 더 높은 다른 인스턴스 한 개를 시작합니다.

- 6.1. 루프를 사용하여 **sha1sum /dev/zero &** 명령의 인스턴스 세 개를 시작합니다.

```
[student@servera ~]$ for i in {1..3}; do sha1sum /dev/zero & done
[1] 1207
[2] 1208
[3] 1209
```

6.2. **nice** 명령을 사용하여 nice 수준이 10인 네 번째 인스턴스를 시작합니다.

```
[student@servera ~]$ nice -n 10 sha1sum /dev/zero &
[4] 1210
```

- ▶ 7. **ps** 및 **pgrep** 명령을 사용하여 각 프로세스의 PID, CPU 사용률, nice 값, 실행 파일 이름을 표시합니다. nice 값이 10인 인스턴스에는 CPU 사용률이 다른 인스턴스보다 낮게 표시됩니다.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
 PID %CPU NI COMMAND
 1207 64.2 0 sha1sum
 1208 65.0 0 sha1sum
 1209 63.9 0 sha1sum
 1210 8.2 10 sha1sum
```

- ▶ 8. **sudo renice** 명령을 사용하여 위 단계의 프로세스 nice 수준을 낮춥니다. nice 수준이 10인 프로세스 인스턴스의 PID 값을 사용하여 nice 수준을 5로 낮춥니다.

```
[student@servera ~]$ sudo renice -n 5 1210
[sudo] password for student:
1210 (process ID) old priority 10, new priority 5
```

- ▶ 9. **ps** 및 **pgrep** 명령을 반복하여 CPU 사용률과 nice 수준을 표시합니다.

```
[student@servera ~]$ ps -o pid,pcpu,nice,comm $(pgrep sha1sum)
 PID %CPU NI COMMAND
 1207 62.9 0 sha1sum
 1208 63.2 0 sha1sum
 1209 63.2 0 sha1sum
 1210 10.9 5 sha1sum
```

- ▶ 10. **pkill** 명령을 사용하여 이름 패턴이 **sha1sum**인 실행 중인 프로세스를 모두 종료합니다.

```
[student@servera ~]$ pkill sha1sum
...output omitted...
```

- ▶ 11. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```



중요

이 연습을 종료하기 전에 모든 연습 프로세스를 종료했는지 확인합니다.

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish tuning-procscheduling
```

이것으로 섹션을 완료합니다.

▶ 랩

시스템 성능 튜닝

이 랩에서는 특정 튜닝 프로파일을 적용하고 CPU 사용률이 높은 기존 프로세스의 예약 우선순위를 조정합니다.

결과

- 컴퓨터 시스템에 대해 특정 튜닝 프로파일을 활성화합니다.
- 프로세스의 CPU 예약 우선순위를 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start tuning-review
```



중요

이 랩에서는 의도적으로 상당한 CPU 리소스를 사용하면서 장치 파일에 무한 체크섬을 수행하는 명령을 사용합니다.

지침

- serverb** 시스템의 현재 튜닝 프로파일을 일반 비전문 튜닝 프로파일인 **balanced** 프로파일로 변경합니다. 현재 튜닝 프로파일이 **balanced** 튜닝 프로파일인 경우 이에 대한 정보를 나열합니다.
- serverb**의 두 프로세스가 높은 CPU 사용률을 보이고 있습니다. 각 프로세스의 **nice** 수준을 10으로 조정하여 다른 프로세스에 더 많은 CPU 시간을 허용합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade tuning-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish tuning-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

시스템 성능 튜닝

이 랩에서는 특정 튜닝 프로파일을 적용하고 CPU 사용률이 높은 기존 프로세스의 예약 우선순위를 조정합니다.

결과

- 컴퓨터 시스템에 대해 특정 튜닝 프로파일을 활성화합니다.
- 프로세스의 CPU 예약 우선순위를 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start tuning-review
```



중요

이 랩에서는 의도적으로 상당한 CPU 리소스를 사용하면서 장치 파일에 무한 체크섬을 수행하는 명령을 사용합니다.

지침

- serverb** 시스템의 현재 튜닝 프로파일을 일반 비전문 튜닝 프로파일인 **balanced** 프로파일로 변경합니다. 현재 튜닝 프로파일이 **balanced** 튜닝 프로파일인 경우 이에 대한 정보를 나열합니다.

- 1.1. **student** 사용자로 **serverb** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. **tuned** 패키지가 설치되었는지 확인합니다.

```
[student@serverb ~]$ dnf list tuned
...output omitted...
Installed Packages
tuned.noarch           2.18.0-1.el9          @System
```

- 1.3. **tuned** 서비스 상태를 확인합니다.

```
[student@serverb ~]$ systemctl is-active tuned
active
```

- 1.4. 사용 가능한 튜닝 프로파일과 해당 설명을 모두 나열합니다. 현재 활성화된 프로파일은 **virtual-guest**입니다.

```
[student@serverb ~]$ sudo tuned-adm list
[sudo] password for student: student
Available profiles:
- accelerator-performance      - Throughput performance based tuning with disabled
                                higher latency STOP states
- balanced                     - General non-specialized tuned profile
- desktop                      - Optimize for the desktop use-case
- hpc-compute                  - Optimize for HPC compute workloads
- intel-sst                     - Configure for Intel Speed Select Base Frequency
- latency-performance          - Optimize for deterministic performance at the cost
                                of increased power consumption
- network-latency              - Optimize for deterministic performance at the cost
                                of increased power consumption, focused on low
                                latency network performance
- network-throughput           - Optimize for streaming network throughput, generally
                                only necessary on older CPUs or 40G+ networks
- optimize-serial-console     - Optimize for serial console use.
- powersave                    - Optimize for low power consumption
- throughput-performance       - Broadly applicable tuning that provides excellent
                                performance across a variety of common server
                                workloads
- virtual-guest                - Optimize for running inside a virtual guest
- virtual-host                 - Optimize for running KVM guests
Current active profile: virtual-guest
```

- 1.5. 현재 활성화된 튜닝 프로파일을 **balanced** 프로파일로 변경합니다.

```
[student@serverb ~]$ sudo tuned-adm profile balanced
```

- 1.6. 현재 활성 조정 된 프로파일의 요약 정보를 나열합니다. 활성 프로파일이 **balanced** 프로파일인지 확인합니다.

```
[student@serverb ~]$ sudo tuned-adm profile_info
Profile name:
balanced

Profile summary:
General non-specialized tuned profile
...output omitted...
```

2. **serverb**의 두 프로세스가 높은 CPU 사용률을 보이고 있습니다. 각 프로세스의 **nice** 수준을 10으로 조정하여 다른 프로세스에 더 많은 CPU 시간을 허용합니다.

- 2.1. **serverb** 시스템에서 CPU 사용률이 높은 두 프로세스를 확인합니다. **ps** 명령은 CPU 사용률이 높은 프로세스를 출력 맨 아래에 나열합니다. CPU 백분율 값은 시스템에 따라 다를 수 있습니다.

```
[student@serverb ~]$ ps aux --sort=pcpu
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
...output omitted...
root     1079 98.5  0.1 225340  2300 ?        RN    06:25   4:29 sha1sum /dev/zero
root     1095 99.0  0.1 225340  2232 ?        R<    06:25   4:30 md5sum /dev/zero
```

2.2. CPU 사용률이 높은 두 프로세스의 현재 **nice** 수준을 각각 식별합니다.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm \
$(pgrep sha1sum;pgrep md5sum)
PID %CPU NI COMMAND
1079 98.8  2 sha1sum
1095 99.1 -2 md5sum
```

2.3. 각 프로세스의 **nice** 수준을 **10**으로 조정합니다. 위 명령 출력에서 프로세스에 올바른 PID 값을 사용합니다.

```
[student@serverb ~]$ sudo renice -n 10 1079 1095
[sudo] password for student:
1079 (process ID) old priority 2, new priority 10
1095 (process ID) old priority -2, new priority 10
```

2.4. 각 프로세스의 현재 **nice** 수준이 10인지 확인합니다.

```
[student@serverb ~]$ ps -o pid,pcpu,nice,comm \
$(pgrep sha1sum;pgrep md5sum)
PID %CPU NI COMMAND
1079 98.9  10 sha1sum
1095 99.2  10 md5sum
```

2.5. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```



중요

이 랩을 종료하기 전에 모든 랩 프로세스를 종료했는지 확인합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade tuning-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish tuning-review
```

이것으로 섹션을 완료합니다.

요약

- 신호는 실행 중인 프로그램에 이벤트를 보고하는 소프트웨어 인터럽트입니다. **kill**, **pkill**, **killall** 명령은 신호를 사용하여 프로세스를 제어합니다.
- 부하 평균은 시스템의 사용량을 추정한 것입니다. 부하 평균 값을 표시하려면 **top**, **uptime** 또는 **w** 명령을 사용할 수 있습니다.
- **tuned** 서비스는 미리 정의된 선택된 튜닝 프로파일에 따라 특정 시스템 요구 사항에 맞게 장치 설정을 자동으로 수정합니다.
- 선택된 프로파일의 모든 변경 사항을 시스템 설정으로 되돌리려면 다른 프로파일로 전환하거나 **tuned** 서비스를 비활성화합니다.
- 시스템이 프로세스에 상대 우선 순위를 할당하여 CPU 액세스를 결정합니다. 이 우선 순위를 프로세스의 **nice** 값이라고 합니다.
- **nice** 명령은 프로세스가 시작될 때 우선 순위를 할당합니다.
- **renice** 명령은 실행 중인 프로세스의 우선순위를 수정합니다.

7장

향후 작업 예약

목적

향후 자동으로 실행되는 작업을 예약합니다.

목표

- 사용자의 crontab 파일을 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.
- 시스템 crontab 파일과 디렉터리를 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.
- systemd 타이머를 활성화 및 비활성화하고 임시 파일을 관리하는 타이머를 구성합니다.

섹션

- 반복 실행 사용자 작업 예약(안내에 따른 연습)
- 반복 실행 시스템 작업 예약(안내에 따른 연습)
- 임시 파일 관리(안내에 따른 연습)

반복 실행 사용자 작업 예약

목표

사용자의 **crontab** 파일을 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.

반복 실행 사용자 작업 설명

반복 실행 작업은 반복적으로 실행되도록 예약됩니다. Red Hat Enterprise Linux 시스템은 기본적으로 활성화되고 시작되는 **crond** 데몬을 제공합니다. **crond** 데몬은 사용자마다 하나의 구성 파일과 시스템 전체에 적용되는 파일 집합 등 여러 구성 파일을 읽습니다. 각 사용자에게는 **crontab -e** 명령으로 편집하는 개인 파일이 있습니다. 반복 실행 작업을 실행할 때 이러한 구성 파일은 사용자와 관리자에게 세부 제어 기능을 제공합니다. 예약된 작업이 리디렉션을 사용하도록 작성되지 않은 경우 **crond** 데몬은 생성된 출력 또는 오류를 작업 소유자에게 이메일로 보냅니다.

반복 실행 사용자 작업 예약

crontab 명령을 사용하여 예약된 작업을 관리합니다. 다음 목록은 로컬 사용자가 작업을 관리하는 데 사용할 수 있는 명령을 보여줍니다.

crontab 명령의 예

명령	용도
crontab -l	현재 사용자의 작업을 나열합니다.
crontab -r	현재 사용자의 모든 작업을 제거합니다.
crontab -e	현재 사용자의 작업을 편집합니다.
crontab filename	모든 작업을 제거하고 filename에서 읽은 작업으로 바꿉니다. 이 명령은 파일이 지정되지 않은 경우 stdin 입력을 사용합니다.

권한 있는 사용자는 **crontab** 명령 **-u** 옵션을 사용하여 다른 사용자의 작업을 관리할 수 있습니다.

crontab 명령은 시스템 작업을 관리하는 데 사용되지 않으며 **root**로 실행되도록 구성된 개인 작업을 악용할 수 있어 **crontab** 명령을 **root** 사용자로 사용하지 않는 것이 좋습니다. 이러한 권한 있는 작업은 반복 실행 시스템 작업을 설명하는 뒷부분의 섹션에 설명된 대로 구성합니다.

사용자 작업 형식 설명

crontab -e 명령은 **EDITOR** 환경 변수가 다른 편집기에 대해 설정되어 있지 않은 경우 기본적으로 **vim** 편집기를 호출합니다. 각 작업은 **crontab** 파일에서 고유한 행을 사용해야 합니다. 반복 실행 작업 작성 시 유효한 항목은 다음 권장 사항을 따르십시오.

- 빈 행은 가독성을 위한 것입니다
- 숫자 기호(#로 시작하는 행은 주석입니다
- **NAME=value** 형식의 환경 변수는 선언된 행 뒤의 모든 행에 영향을 미칩니다

표준 변수 설정에는 **crontab** 파일의 나머지 행을 해석하는 데 사용되는 쉘을 선언하는 **SHELL** 변수가 포함됩니다. **MAILTO** 변수는 이메일로 출력을 받을 사람을 결정합니다.

**참고**

이메일 기능을 사용하려면 로컬 메일 서버 또는 SMTP 릴레이에 대한 추가 시스템 구성이 필요합니다.

crontab 파일의 필드는 다음 순서로 나타납니다.

- 년
- 시간
- 일
- 월
- 요일
- 명령

이 명령은 일 또는 요일 필드가 * 문자 이외의 동일한 값을 사용하는 경우 실행됩니다. 예를 들어 매월 11일과 매주 금요일 12:15(24시간 형식)에 명령을 실행하려면 다음 작업 형식을 사용합니다.

```
15 12 11 * Fri command
```

처음 5개 필드는 모두 동일한 구문 규칙을 사용합니다.

- * 문자를 사용하여 필드의 가능한 모든 인스턴스에서 실행합니다.
- 분 또는 시간, 날짜 또는 요일을 지정하는 숫자입니다. 요일의 경우 0은 일요일, 1은 월요일, 2는 화요일에 해당합니다. 7도 일요일에 해당합니다.
- x 및 y 값이 포함되는 범위의 경우 x-y를 사용합니다.
- 목록에는 x, y를 사용합니다. 범위도 목록에 포함할 수 있습니다. 예를 들어 Minutes 열에 5, 10-13, 17을 입력하면 작업이 정시 5분 후, 10분 후, 11분 후, 12분 후, 13분 후, 17분 후에 실행됩니다.
- */x는 x만큼의 간격을 나타냅니다. 예를 들어, Minutes 열에 */7을 입력하면 7분마다 작업이 실행됩니다.

또한 월과 요일에 Jan, Feb, Mon, Tue와 같은 3자로 된 영어 약자를 사용할 수 있습니다.

마지막 필드에는 기본 헬을 사용하여 실행할 전체 명령과 옵션 및 인수가 포함됩니다. 명령에 이스케이프 처리되지 않은 백분율 기호(%)가 포함된 경우 해당 백분율 기호는 줄 바꿈으로 처리되며, 백분율 기호 다음의 모든 내용이 명령에 **stdin** 입력으로 전달됩니다.

반복 실행 사용자 작업의 예

다음 작업은 매년 2월 3일 오전 9시 정각에 **/usr/local/bin/yearly_backup** 명령을 실행합니다. 2월은 해당 연도의 두 번째 달이므로 예제에서 숫자 2로 표시됩니다.

```
0 9 3 2 * /usr/local/bin/yearly_backup
```

다음 작업은 7월 매주 금요일 오전 9시부터 오후 4시까지 5분마다 이 작업의 소유자에게 **Chime**이라는 단어가 포함된 이메일을 전송합니다.

```
*/5 9-16 * Jul 5 echo "Chime"
```

앞의 **9-16** 시간 범위는 작업 타이머가 9번째 시간(09:00)에 시작되어 16번째 시간이 끝날 때까지(16:59) 계속됨을 의미합니다. 작업은 **09:00**에 처음 실행되고 **16:55**에 마지막으로 실행됩니다. **16:55**의 5분 후는 **17:00**인데 이는 지정된 시간 범위를 벗어나기 때문입니다.

범위가 단일 값이 아니라 특정 시간에 해당하는 경우 범위 내의 모든 시간이 일치합니다. 이 예제에서는 시간이 **9-16**이므로 09:00부터 16:55까지 5분마다 일치합니다.



참고

이 작업 예제에서는 출력을 이메일로 전송합니다. **crond**이 출력을 리디렉션 없이 **STDIO** 채널로 이동할 수 있도록 허용된 작업임을 인식하기 때문입니다. cron 작업은 출력 장치(제어 터미널이라고 함) 없이 백그라운드 환경에서 실행되므로 **crond**는 출력을 버퍼링하고 구성에 지정된 사용자에게 보낼 이메일을 생성합니다. 시스템 작업의 경우 이메일이 **root** 계정으로 전송됩니다.

다음 작업은 (월요일 ~ 금요일) 매일 자정 2분 전에 **/usr/local/bin/daily_report** 명령을 실행합니다.

```
58 23 * * 1-5 /usr/local/bin/daily_report
```

다음 작업은 모든 업무일(월요일 ~ 금요일) 오전 9시에 메일 메시지 **Checking in**을 수신자 **developer@example.com**에게 보내는 **mutt** 명령을 실행합니다.

```
0 9 * * 1-5 mutt -s "Checking in" developer@example.com % Hi there, just checking in.
```



참조

crond(1), **crontab(5)** 및 **crontab(8)** 도움말 페이지

▶ 연습 가이드

반복 실행 사용자 작업 예약

이 연습에서는 **crontab** 명령을 사용하여 권한이 없는 사용자로 반복 스케줄에 따라 실행되도록 명령을 예약합니다.

결과

- 권한이 없는 사용자로 실행되도록 반복 실행 작업을 예약합니다.
- 예약된 반복 실행 작업이 실행하는 명령을 검사합니다.
- 예약된 반복 실행 작업을 제거합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start scheduling-cron
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 2분마다 현재 날짜와 시간을 **/home/student/my_first_cron_job.txt** 파일에 추가하는 **student** 사용자로 반복 실행 작업을 예약합니다. **date** 명령을 사용하여 현재 날짜와 시간을 표시합니다. 작업은 현재 시간 하루 전부터 하루 후까지만 실행해야 합니다. 작업을 다른 날 실행하면 안 됩니다.

- 2.1. **date** 명령을 사용하여 현재 날짜와 시간을 표시합니다. 다음 단계를 수행해야 하는 요일을 기록해 둡니다.

```
[student@workstation ~]$ date
Wed Mar 15 07:33:01 PM EDT 2023
[student@servera ~]$
```

**참고**

`date -d "last day" +%a` 명령을 사용하여 현재 시간 이전 날짜를 표시하고 `date -d "next day" +%a` 명령을 사용하여 현재 시간 이후 날짜를 표시할 수 있습니다.

```
[student@servera ~]$ date -d "last day" +%a
Tue
[student@servera ~]$ date -d "next day" +%a
Thu
```

2.2. 기본 텍스트 편집기에서 `crontab` 파일을 엽니다.

```
[student@servera ~]$ crontab -e
```

2.3. 다음 행을 삽입합니다. 현재 시간 하루 전에서 하루 후까지의 일 범위를 교체합니다.

```
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

2.4. `Esc`를 누르고 `:wq`를 입력하여 변경 사항을 저장하고 편집기를 종료합니다. 편집기가 종료되면 다음과 같은 출력이 표시됩니다.

```
...output omitted...
crontab: installing new crontab
[student@servera ~]$
```

- ▶ 3. `crontab -l` 명령을 사용하여 예약된 반복 실행 작업을 나열합니다. 위 단계에서 반복 실행 작업으로 실행되도록 예약한 명령을 검사합니다.

작업이 `/usr/bin/date` 명령을 실행하고 `/home/student/my_first_cron_job.txt` 파일에 출력을 추가하는지 확인합니다.

```
[student@servera ~]$ crontab -l
*/2 * * * Tue-Thu /usr/bin/date >> /home/student/my_first_cron_job.txt
```

- ▶ 4. 예약한 반복 실행 작업의 성공적인 실행 결과로 `/home/student/my_first_cron_job.txt` 파일이 생성될 때까지 쉘 프롬프트가 유휴 상태가 될 것을 지시합니다. 쉘 프롬프트가 반환될 때까지 기다립니다.

`while` 명령은 `! test -f`를 사용하여 반복문을 계속 실행하고 `my_first_cron_job.txt` 파일이 `/home/student` 디렉터리에 생성될 때까지 1초 동안 유휴 상태가 됩니다.

```
[student@servera ~]$ while ! test -f my_first_cron_job.txt; do sleep 1s; done
```

- ▶ 5. `/home/student/my_first_cron_job.txt` 파일의 내용이 `date` 명령의 출력과 일치하는지 확인합니다.

```
[student@servera ~]$ cat my_first_cron_job.txt
Wed Mar 15 07:40:01 PM EDT 2023
```

▶ 6. **student**에 대해 예약된 반복 실행 작업을 모두 제거합니다.

6.1. **student**에 대해 예약된 반복 실행 작업을 모두 제거합니다.

```
[student@servera ~]$ crontab -r
```

6.2. **student** 사용자에 대한 반복 실행 작업이 없는지 확인합니다.

```
[student@servera ~]$ crontab -l
no crontab for student
```

6.3. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish scheduling-cron
```

이것으로 섹션을 완료합니다.

반복 실행 시스템 작업 예약

목표

시스템 **crontab** 파일과 디렉터리를 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.

반복 실행 시스템 작업

시스템 관리자는 종종 반복 실행 작업을 실행해야 합니다. 사용자 계정이 아닌 시스템 계정에서 이러한 작업을 실행하는 것이 좋습니다. **crontab** 명령 대신 시스템 전체 crontab 파일을 사용하여 이러한 작업을 예약합니다. 시스템 전체 crontab 파일의 작업 항목은 사용자의 crontab 항목과 유사합니다. 시스템 전체 crontab 파일에는 명령을 실행하는 사용자를 지정하는 명령 필드 앞에 추가 필드가 있습니다.

/etc/crontab 파일의 주석에는 구문 다이어그램이 있습니다.

```

SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# Example of job definition:
# ----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .---- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .--- day of week (0 - 6) (Sunday=0 or 7) OR
# | | | | sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed

```

/etc/cron.d/ 디렉터리의 **/etc/crontab** 파일과 기타 파일은 반복 실행 시스템 작업을 정의합니다. 반복 실행 시스템 작업을 예약하려면 항상 **/etc/cron.d/** 디렉터리에 사용자 지정 crontab 파일을 생성합니다. 패키지 업데이트가 **/etc/crontab** 파일을 덮어쓰지 않도록 사용자 지정 crontab 파일을 **/etc/cron.d** 디렉터리에 저장합니다. 반복 실행 시스템 작업이 필요한 패키지는 작업 항목이 포함된 **/etc/cron.d/** 디렉터리에 해당 crontab 파일을 저장합니다. 또한 관리자는 이 위치를 사용하여 관련 작업을 하나의 파일로 그룹화합니다.

crontab 시스템에는 매시간, 매일, 매주, 매월 실행해야 하는 스크립트의 리포지토리도 포함되어 있습니다. 이러한 리포지토리는 **/etc/cron.hourly/**, **/etc/cron.daily/**, **/etc/cron.weekly/**, **/etc/cron.monthly/** 디렉터리에 배치됩니다. 해당 디렉터리에는 crontab 파일이 아닌 실행 가능한 쉘 스크립트가 들어 있습니다.



참고

스크립트를 실행 파일로 만들려면 **chmod +x script_name** 명령을 사용합니다. 스크립트를 실행하려면 실행 가능해야 합니다.

Anacron을 사용하여 정기 명령 실행

또한 `run-parts` 명령은 `/etc/anacrontab` 구성 파일의 작업을 매일, 매주, 매월 실행합니다.

`/etc/anacrontab` 파일을 사용하면 예약된 작업이 항상 실행되고 시스템이 꺼져 있거나 최대 절전 모드로 전환되어 실수로 건너뛰지 않습니다. 예를 들어 매일 실행되는 시스템 작업이 시스템 재부팅으로 인해 지정된 시간에 실행되지 않은 경우 시스템이 준비되면 작업이 완료됩니다. `/etc/anacrontab` 파일의 `Delay in minutes` 매개 변수에 지정된 경우 작업이 시작되기 전에 자연이 발생할 수 있습니다.

`/var/spool/anacron/` 디렉터리의 파일에 따라 일별, 주별, 월별 작업이 결정됩니다. `cron` 데몬은 `/etc/anacrontab` 파일의 작업을 시작할 때 해당 파일의 타임스탬프를 업데이트합니다. 이 타임스탬프를 사용하여 작업이 마지막으로 실행된 시간을 확인할 수 있습니다. `/etc/anacrontab` 파일의 구문은 일반적인 `crontab` 구성 파일과 다릅니다. `/etc/anacrontab` 파일에는 행당 4개의 필드가 다음과 같이 포함되어 있습니다.

Period in days

반복 스케줄에 따라 실행되는 작업의 간격(일)을 정의합니다. 이 필드에는 정수 또는 매크로 값을 사용합니다. 예를 들어 매크로 `@daily`는 정수 1과 동일하며, 작업을 매일 실행합니다. 마찬가지로 매크로 `@weekly`는 정수 7과 동일하며, 작업을 매주 실행합니다.

Delay in minutes

작업을 시작하기 전에 `cron` 데몬이 기다려야 하는 시간을 정의합니다.

Job identifier

로그 메시지에서 고유한 작업 이름을 식별합니다.

Command

실행할 명령입니다.

`/etc/anacrontab` 파일에는 `NAME=value` 구문을 사용하는 환경 변수 선언도 포함되어 있습니다. `START_HOURS_RANGE` 변수는 작업이 실행되는 시간 간격을 지정합니다. 이 범위를 벗어난 작업은 시작되지 않습니다. 작업이 특정 날짜에 이 시간 간격 내로 실행되지 않으면 다음 날 실행될 때까지 기다려야 합니다.

systemd 타이머

`systemd` 타이머 유닛은 유닛 이름이 타이머 유닛 이름과 일치하는 다른 유형(예: 서비스)의 또 다른 유닛을 활성화합니다. 타이머 장치는 timer-based 다른 유닛의 활성화. 편리한 디버깅을 위해 `systemd` 타이머 유닛은 시스템 저널에 타이머 이벤트를 기록합니다.

샘플 타이머 장치

`sysstat` 패키지는 10분마다 시스템 통계를 수집하는 `sysstat-collect.timer` 서비스라는 `systemd` 타이머 유닛을 제공합니다. 다음 출력에는 `/usr/lib/systemd/system/sysstat-collect.timer` 구성 파일의 콘텐츠가 표시되어 있습니다.

```
...output omitted...
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

OnCalendar=*:00/10 옵션은 이 타이머 유닛이 10분마다 해당 `sysstat-collect.service` 유닛을 활성화함을 나타냅니다. 더 복잡한 시간 간격을 지정할 수도 있습니다.

예를 들어 `OnCalendar` 옵션에 대해 값 `2022-04-* 12:35,37,39:16`을 지정하면 타이머 유닛에서 2022년 4월 동안 매일 `12:35:16, 12:37:16, 12:39:16`에 해당 서비스 유닛을 활성화합니다. `OnUnitActiveSec` 옵션을 사용하여 상대 타이머를 지정할 수도 있습니다. 예를 들어 `OnUnitActiveSec=15min` 옵션을 사용하면 타이머 유닛에서 해당 유닛을 마지막으로 활성화하고 15분 후에 해당 유닛이 시작되도록 트리거합니다.



중요

`/usr/lib/systemd/system` 디렉터리 아래의 구성 파일에서 유닛을 수정하지 마십시오. `systemd` 유닛이 해당 파일의 구성 변경 사항을 재정의하기 때문입니다. `/etc/systemd/system` 디렉터리에 구성 파일의 복사본을 생성한 다음, 복사된 파일을 수정하여 프로바이더 패키지 업데이트가 변경 사항을 재정의하지 않도록 합니다. 같은 이름의 두 파일이 `/usr/lib/systemd/system` 및 `/etc/systemd/system` 디렉터리에 있는 경우 `systemd` 타이머 유닛은 `/etc/systemd/system` 디렉터리의 파일을 구문 분석합니다.

타이머 유닛 구성 파일을 변경한 후에는 `systemctl daemon-reload` 명령을 사용하여 `systemd` 타이머 유닛이 변경 사항을 로드하는지 확인합니다.

```
[root@host ~]# systemctl daemon-reload
```

`systemd` 데몬 구성은 다시 로드한 후 `systemctl` 명령을 사용하여 타이머 유닛을 활성화합니다.

```
[root@host ~]# systemctl enable --now <unitname>.timer
```



참조

`crontab(5)`, `anacron(8)`, `anacrontab(5)`, `systemd.time(7)`, `systemd.timer(5)` 및 `crond(8)` 도움말 페이지

▶ 연습 가이드

반복 실행 시스템 작업 예약

이 연습에서는 시스템 crontab 디렉터리에 구성 파일을 추가하여 다양한 스케줄에 따라 실행되도록 명령을 예약합니다.

결과

- 반복 실행 시스템 작업을 예약하여 활성 사용자 수를 계산합니다.
- 시스템 활동 데이터를 수집하는 **systemd** 타이머 장치를 업데이트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start scheduling-system
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 시스템의 현재 활성 사용자 수를 나타내는 로그 메시지를 생성하는 반복 실행 시스템 작업을 예약합니다. 이 작업은 매일 실행하고 **w -h | wc -l** 명령을 사용하여 시스템의 활성 사용자 수를 검색해야 합니다. **logger** 명령을 사용하여 현재 활성 사용자의 로그 메시지를 생성합니다.

2.1. 다음 내용으로 **/etc/cron.daily/usercount** 스크립트 파일을 생성합니다.

```
#!/bin/bash
USERCOUNT=$(w -h | wc -l)
logger "There are currently ${USERCOUNT} active users"
```

2.2. 스크립트 파일을 실행 파일로 만듭니다.

```
[root@servera ~]# chmod +x /etc/cron.daily/usercount
```

- ▶ 3. **sysstat** 패키지를 설치합니다. 타이머 유닛은 **/usr/lib64/sa/sa1** 쉘 스크립트로 시스템 활동 데이터를 수집하도록 서비스 유닛을 10분마다 트리거해야 합니다. 2분마다 시스템 활동 데이터를 수집하도록 타이머 유닛 구성 파일을 변경합니다.

- 3.1. **sysstat** 패키지를 설치합니다.

```
[root@servera ~]# dnf install sysstat
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 3.2. **/usr/lib/systemd/system/sysstat-collect.timer** 파일을 **/etc/systemd/system/sysstat-collect.timer** 파일에 복사합니다.

```
[root@servera ~]# cp /usr/lib/systemd/system/sysstat-collect.timer \
/etc/systemd/system/sysstat-collect.timer
```

- 3.3. 타이머 유닛이 2분마다 실행되도록 **/etc/systemd/system/sysstat-collect.timer** 파일을 편집합니다. 주석 처리된 행에 있는 경우를 포함하여 유닛 구성 파일 전체에 걸쳐 모든 **10 minutes**라는 문자열 항목을 **2 minutes**로 바꿉니다. **vim /etc/systemd/system/sysstat-collect.timer** 명령을 사용하여 구성 파일을 편집합니다.

이러한 변경 사항에서 **sysstat-collect.timer** 유닛은 **sysstat-collect.service** 유닛을 2분마다 트리거하고 **/var/log/sa** 디렉터리의 바이너리 파일에 시스템 활동 데이터를 수집합니다.

```
...output omitted...
# Activates activity collector every 2 minutes

[Unit]
Description=Run system activity accounting tool every 2 minutes

[Timer]
OnCalendar=*:00/2

[Install]
WantedBy=sysstat.service
```

- 3.4. **systemd** 데몬에 변경 사항을 알립니다.

```
[root@servera ~]# systemctl daemon-reload
```

- 3.5. **sysstat-collect.timer** 유닛을 활성화합니다.

```
[root@servera ~]# systemctl enable --now sysstat-collect.timer
...output omitted...
```

- 3.6. 바이너리 파일이 **/var/log/sa** 디렉터리에 생성될 때까지 기다립니다.

while 명령, **ls /var/log/sa | wc -l**는 파일이 없으면 0을 반환하고 파일이 있으면 1을 반환합니다. 파일이 없으면 **while** 명령이 1초 동안 일시 중지됩니다. 파일이 있으면 **while** 반복문이 종료됩니다.

```
[root@servera ~]# while [ $(ls /var/log/sa | wc -l) -eq 0 ]; \
do sleep 1s; done
```

3.7. `/var/log/sa` 디렉터리의 바이너리 파일이 2분 이내에 수정되었는지 확인합니다.

```
[root@servera ~]# ls -l /var/log/sa
total 4
-rw-r--r-- 1 root root 2540 Apr  5 04:08 sa05
[root@servera ~]# date
Tue Apr  5 04:08:29 AM EDT 2022
```

3.8. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish scheduling-system
```

이것으로 섹션을 완료합니다.

임시 파일 관리

목표

systemd 타이머를 활성화 및 비활성화하고 임시 파일을 관리하는 타이머를 구성합니다.

임시 파일 관리

대부분의 중요한 애플리케이션과 서비스는 임시 파일과 디렉터리를 사용합니다. 일부 애플리케이션 및 사용자는 **/tmp** 디렉터리를 사용하여 임시 작업 데이터를 보관하지만 기타 애플리케이션은 **/run** 아래에 있는데 몬 및 사용자 지정 휘발성 디렉터리와 같이 메모리에만 있는 작업별 위치를 사용합니다. 시스템이 재부팅되거나 전원이 꺼지면 메모리 기반 파일 시스템이 자동으로 정리됩니다.

일반적으로 데몬과 스크립트는 예상되는 임시 파일과 디렉터리가 있는 경우에만 정확하게 작동합니다. 또한 디스크 공간 문제나 부실 작업 데이터가 발생하지 않도록 영구저장장치에 있는 임시 파일을 제거해야 합니다.

Red Hat Enterprise Linux에는 **systemd-tmpfiles** 툴이 포함되어 있습니다. 이 툴은 임시 디렉터리와 파일을 관리하는 체계적이고 구성 가능한 방법을 제공합니다.

시스템 부팅 시 처음으로 시작되는 **systemd** 서비스 유닛 중 하나는 **systemd-tmpfiles-setup** 서비스입니다. 이 서비스는 **/usr/lib/tmpfiles.d/* .conf**, **/run/tmpfiles.d/* .conf**, **/etc/tmpfiles.d/* .conf** 구성 파일에서 지침을 읽는 **systemd-tmpfiles** 명령 **--create** **--remove** 옵션을 실행합니다. 이러한 구성 파일은 권한을 사용하여 생성, 삭제 또는 보호하도록 **systemd-tmpfiles-setup** 서비스에 지시하는 파일 및 디렉터리를 나열합니다.

Systemd 타이머를 사용하여 임시 파일 정리

장기 실행 시스템이 부실 데이터로 디스크를 채우지 않도록 하기 위해 **systemd-tmpfiles-clean.timer**라는 **systemd** 타이머 유닛이 일정 간격으로 **systemd-tmpfiles-clean.service** 유닛을 트리거하며, 이 유닛은 다시 **systemd-tmpfiles --clean** 명령을 실행합니다.

systemd 타이머 유닛 구성에는 타이머와 이름이 같은 서비스를 시작하는 방법을 나타내는 **[Timer]** 섹션이 있습니다.

다음 **systemctl** 명령을 사용하여 **systemd-tmpfiles-clean.timer** 장치 구성 파일의 내용을 봅니다.

```
[user@host ~]$ systemctl cat systemd-tmpfiles-clean.timer
# /usr/lib/systemd/system/systemd-tmpfiles-clean.timer
# SPDX-License-Identifier: LGPL-2.1-or-later
#
# This file is part of systemd.
#
# systemd is free software; you can redistribute it and/or modify it
# under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation; either version 2.1 of the License, or
# (at your option) any later version.

[Unit]
Description=Daily Cleanup of Temporary Directories
```

```
Documentation=man:tmpfiles.d(5) man:systemd-tmpfiles(8)
ConditionPathExists=!/etc/initrd-release

[Timer]
OnBootSec=15min
OnUnitActiveSec=1d
```

위 구성에서 **OnBootSec=15min** 매개 변수는 **systemd-tmpfiles-clean.service** 유닛이 시스템이 부팅되고 15분 후에 트리거됨을 나타냅니다. **OnUnitActiveSec=1d** 매개 변수는 **systemd-tmpfiles-clean.service** 유닛에 대한 추가 트리거가 서비스 유닛이 마지막으로 활성화되고 24시간 후에 발생함을 나타냅니다.

요구 사항에 따라 **systemd-tmpfiles-clean.timer** 유닛 구성 파일의 매개 변수를 변경하십시오. 예를 들어 **OnUnitActiveSec** 매개 변수에 값 **30min**을 사용하면 서비스 유닛이 마지막으로 활성화되고 30분 후에 **systemd-tmpfiles-clean.service** 유닛이 트리거됩니다. 결과적으로 변경 사항을 인식한 후 30분마다 **systemd-tmpfiles-clean.service** 유닛이 트리거됩니다.

타이머 유닛 구성 파일을 변경한 후에는 **systemd** 데몬에서 새 구성을 로드하도록 **systemctl daemon-reload** 명령을 사용합니다.

```
[root@host ~]# systemctl daemon-reload
```

수동으로 임시 파일 정리

systemd-tmpfiles --clean 명령은 **systemd-tmpfiles --create** 명령과 동일한 구성 파일을 구문 분석하지만 파일과 디렉터리를 생성하는 대신 구성 파일에 정의된 최대 사용 기간보다 최근에 액세스, 변경 또는 수정되지 않은 모든 파일을 삭제합니다.

systemd-tmpfiles 서비스의 구성 파일 형식에 대한 자세한 내용은 **tmpfiles.d(5)** 도움말 페이지에서 확인할 수 있습니다. 구문은 Type, Path, Mode, UID, GID, Age, Argument 열로 구성됩니다. Type은 **systemd-tmpfiles** 서비스에서 수행해야 하는 작업을 가리킵니다. 예를 들어 **d** 는 디렉터리가 없을 경우 디렉터리를 생성하고, **Z** 는 SELinux 컨텍스트와 파일 권한 및 소유권을 재귀적으로 복구합니다.

다음은 명령은 설명이 포함된 구성을 제거합니다.

```
d /run/systemd/seats 0755 root root -
```

파일과 디렉터리를 만들 경우 **root** 사용자와 **root** 그룹이 소유자인 **/run/systemd/seats** 디렉터리가 없는 경우 해당 디렉터리를 만들고 **rwxr-xr-x** 권한을 부여합니다. 이 디렉터리가 있으면 아무 작업도 수행하지 않습니다. **systemd-tmpfiles** 서비스는 이 디렉터리를 자동으로 제거하지 않습니다.

```
D /home/student 0700 student student 1d
```

/home/student 디렉터리가 없는 경우 해당 디렉터리를 만듭니다. 해당 디렉터리가 있는 경우 모든 콘텐츠를 제거합니다. 시스템에서 **systemd-tmpfiles --clean** 명령을 실행하면 디렉터리에서 1일 초과 액세스, 변경 또는 수정되지 않은 모든 파일이 제거됩니다.

```
L /run/fstablink - root root - /etc/fstab
```

/etc/fstab 디렉터리를 가리키는 **/run/fstablink** 심볼릭 링크를 만듭니다. 이 행을 자동으로 삭제하지 마십시오.

구성 파일 우선순위

`systemd-tmpfiles-clean` 서비스 구성 파일은 다음 세 위치에 존재할 수 있습니다.

- `/etc/tmpfiles.d/*.conf`
- `/run/tmpfiles.d/*.conf`
- `/usr/lib/tmpfiles.d/*.conf`

`/etc/tmpfiles.d/` 디렉터리의 파일은 사용자 지정 임시 위치를 구성하고 벤더가 제공한 기본값을 재정의하는데 사용합니다. `/run/tmpfiles.d/` 디렉터리의 파일은 일반적으로 데몬이 자체 런타임 임시 파일을 관리하는데 사용하는 휘발성 파일입니다. 관련 RPM 패키지에서 `/usr/lib/tmpfiles.d/` 디렉터리에 파일을 제공하므로 이러한 파일을 편집하지 마십시오.

`/run/tmpfiles.d/` 디렉터리의 파일 이름이 `/usr/lib/tmpfiles.d/` 디렉터리의 파일 이름과 같은 경우 서비스는 `/run/tmpfiles.d/` 디렉터리의 파일을 사용합니다. `/etc/tmpfiles.d/` 디렉터리의 파일 이름이 `/run/tmpfiles.d/` 또는 `/usr/lib/tmpfiles.d/` 디렉터리의 파일 이름과 같은 경우 서비스는 `/etc/tmpfiles.d/` 디렉터리의 파일을 사용합니다.

이러한 우선순위 규칙에 따라 관련 파일을 `/etc/tmpfiles.d/` 디렉터리에 복사하고 편집하여 벤더가 제공한 설정을 재정의할 수 있습니다. 이러한 구성 위치를 정확하게 사용하면 중앙 구성 관리 시스템에서 관리자가 구성한 설정을 관리할 수 있으며, 패키지 업데이트가 구성된 설정을 덮어쓰지 않습니다.



참고

새 구성 또는 수정된 구성을 테스트할 때 한 번에 단일 구성 파일의 명령만 적용하십시오.
`systemd-tmpfiles` 명령줄에 단일 구성 파일의 이름을 지정하십시오.



참조

`systemd-tmpfiles(8)`, `tmpfiles.d(5)`, `stat(1)`, `stat(2)`, `systemd.timer(5)` 도
움말 페이지

▶ 연습 가이드

임시 파일 관리

이 연습에서는 `/tmp` 디렉터리에서 임시 파일을 제거하는 속도를 변경하고 다른 디렉터리에서 파일을 정기적으로 제거하기 위해 `systemd-tmpfiles`를 구성합니다.

결과

- `/tmp` 디렉터리에서 사용되지 않는 임시 파일을 제거하도록 `systemd-tmpfiles`를 구성합니다.
- 다른 디렉터리에서 파일을 정기적으로 제거하도록 `systemd-tmpfiles`를 구성합니다.

시작하기 전에

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start scheduling-tempfiles
```

지침

- ▶ 1. `servera` 시스템에 `student` 사용자로 로그인한 후 `root` 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 지난 5일 동안 사용되지 않은 파일의 `/tmp` 디렉터리를 정리하도록 `systemd-tmpfiles` 서비스를 구성합니다. 패키지 업데이트가 구성 파일을 덮어쓰지 않도록 합니다.

- 2.1. `/usr/lib/tmpfiles.d/tmp.conf` 파일을 `/etc/tmpfiles.d` 디렉터리에 복사합니다.

```
[root@servera ~]# cp /usr/lib/tmpfiles.d/tmp.conf \
/etc/tmpfiles.d/tmp.conf
```

- 2.2. `/etc/tmpfiles.d/tmp.conf` 파일에서 `/tmp` 디렉터리에 적용되는 구성 행을 검색합니다. 해당 구성 행에 있는 임시 파일의 기존 사용 기간을 새 사용 기간인 5일로 바꿉니다. 주석으로 처리된 행을 포함하여 파일의 기타 모든 행을 제거합니다. `vim /etc/tmpfiles.d/tmp.conf` 명령을 사용하여 구성 파일을 편집할 수 있습니다.

위 구성에서 `q` 유형은 `d` 유형과 같고, `/tmp` 디렉터리가 없으면 생성하도록 `systemd-tmpfiles` 서비스에 지시합니다. 디렉터리의 8진수 권한은 `1777`로 설정되어야 합니다. `/tmp` 디렉터리를 소유하는 사용자와 그룹이 둘 다 `root`여야 합니다. `/tmp` 디렉터리에는 지난 5일 동안 사용되지 않은 임시 파일이 포함되지 않아야 합니다.

/etc/tmpfiles.d/tmp.conf 파일은 다음과 같이 표시됩니다.

```
q /tmp 1777 root root 5d
```

- 2.3. /etc/tmpfiles.d/tmp.conf 파일 구성 확인합니다.

명령에서 오류가 반환되지 않았으므로 구성 설정이 올바른 것입니다.

```
[root@servera ~]# systemctl-tmpfiles --clean /etc/tmpfiles.d/tmp.conf
```

- ▶ 3. /run/momentary 디렉터리가 있고 사용자 및 그룹 소유권이 root 사용자로 설정되었는지 확인하는 새 구성을 추가합니다. 디렉터리에 대한 8진수 권한은 0700이어야 합니다. 이 구성은 이 디렉터리에서 지난 30초 동안 사용되지 않은 파일을 모두 제거해야 합니다.

- 3.1. 다음 내용으로 /etc/tmpfiles.d/momentary.conf 파일을 생성합니다.

해당 구성에서는 systemd-tmpfiles 서비스에서 /run/momentary 디렉터리가 있고 8진수 권한이 0700으로 설정되었는지 확인합니다. /run/momentary 디렉터리의 소유권이 root 사용자 및 그룹이어야 합니다. 서비스는 이 디렉터리에서 30초 동안 사용되지 않은 파일을 제거합니다.

```
[root@servera ~]# vim /etc/tmpfiles.d/momentary.conf  
d /run/momentary 0700 root root 30s
```

- 3.2. /etc/tmpfiles.d/momentary.conf 파일 구성 확인합니다. 이 명령은 /run/momentary 디렉터리가 없는 경우 해당 디렉터리를 생성합니다.

명령에서 오류가 반환되지 않았으므로 구성 설정이 올바른 것입니다.

```
[root@servera ~]# systemctl-tmpfiles --create \  
/etc/tmpfiles.d/momentary.conf
```

- 3.3. systemd-tmpfiles 명령이 적절한 권한, 소유자, 그룹 소유자를 사용하여 /run/momentary 디렉터리를 생성하는지 확인합니다.

/run/momentary 디렉터리의 8진수 권한이 0700으로 설정되어 있고 사용자 및 그룹 소유권이 root로 설정되어 있습니다.

```
[root@servera ~]# ls -ld /run/momentary  
drwx----- 2 root root 40 Apr 4 06:35 /run/momentary
```

- ▶ 4. systemctl-tmpfiles --clean 명령이 /run/momentary 디렉터리에 대한 systemd-tmpfiles 구성에 따라 디렉터리에서 지난 30초 동안 사용되지 않은 파일을 모두 제거하는지 확인합니다.

- 4.1. /run/momentary/test 파일을 생성합니다.

```
[root@servera ~]# touch /run/momentary/test
```

- 4.2. 30초 동안 반환되지 않도록 쉘 프롬프트를 구성합니다.

```
[root@servera ~]# sleep 30
```

- 4.3. 쉘 프롬프트가 반환되면 `/etc/tmpfiles.d/momentary.conf` 구성 파일의 참조 규칙에 따라 `/run/momentary` 디렉터리에서 부실 파일을 정리합니다.
- `/run/momentary/test` 파일은 30초 동안 사용되지 않은 상태로 유지되었으므로 이 명령에서 제거합니다. 이러한 동작은 `/etc/tmpfiles.d/momentary.conf` 구성 파일의 참조 규칙을 기반으로 합니다.

```
[root@servera ~]# systemctl-tmpfiles --clean \
/etc/tmpfiles.d/momentary.conf
```

- 4.4. `/run/momentary/test` 파일이 없음을 확인합니다.

```
[root@servera ~]# ls -l /run/momentary/test
ls: cannot access '/run/momentary/test': No such file or directory
```

- 4.5. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish scheduling-tempfiles
```

이것으로 섹션을 완료합니다.

▶ 퀴즈

향후 작업 예약

다음 질문에 대한 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 현재 로그인한 사용자에 대해 예약된 반복 실행 사용자 작업을 모두 표시하는 명령은 무엇입니까?
 a. crontab -r
 b. crontab -l
 c. crontab -u
 d. crontab -v
- ▶ 2. 다음 중 월요일부터 금요일까지 매일 오전 9시부터 오후 6시까지 매시간 /usr/local/bin/daily_backup 명령을 실행하는 작업 형식은 무엇입니까?
 a. 00 * * * Mon-Fri /usr/local/bin/daily_backup
 b. * */9 * * Mon-Fri /usr/local/bin/daily_backup
 c. 00 */18 * * * /usr/local/bin/daily_backup
 d. 00 09-18 * * Mon-Fri /usr/local/bin/daily_backup
- ▶ 3. 다음 중 매일 실행할 쉘 스크립트가 포함된 디렉터리는 무엇입니까?
 a. /etc/cron.d
 b. /etc/cron.hourly
 c. /etc/cron.daily
 d. /etc/cron.weekly
- ▶ 4. 다음 중 매일, 매주, 매월 실행되는 시스템 작업에 대한 설정을 정의하는 구성 파일은 무엇입니까?
 a. /etc/crontab
 b. /etc/anacrontab
 c. /etc/inittab
 d. /etc/sysconfig/crond
- ▶ 5. 다음 중 정기적으로 임시 파일 정리를 트리거하는 systemd 유닛은 무엇입니까?
 a. systemd-tmpfiles-clean.timer
 b. systemd-tmpfiles-clean.service
 c. dnf-makecache.timer
 d. unbound-anchor.timer

▶ 솔루션

향후 작업 예약

다음 질문에 대한 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 현재 로그인한 사용자에 대해 예약된 반복 실행 사용자 작업을 모두 표시하는 명령은 무엇입니까?
 a. crontab -r
 b. crontab -l
 c. crontab -u
 d. crontab -v
- ▶ 2. 다음 중 월요일부터 금요일까지 매일 오전 9시부터 오후 6시까지 매시간 /usr/local/bin/daily_backup 명령을 실행하는 작업 형식은 무엇입니까?
 a. 00 * * * Mon-Fri /usr/local/bin/daily_backup
 b. * */9 * * Mon-Fri /usr/local/bin/daily_backup
 c. 00 */18 * * * /usr/local/bin/daily_backup
 d. 00 09-18 * * Mon-Fri /usr/local/bin/daily_backup
- ▶ 3. 다음 중 매일 실행할 쉘 스크립트가 포함된 디렉터리는 무엇입니까?
 a. /etc/cron.d
 b. /etc/cron.hourly
 c. /etc/cron.daily
 d. /etc/cron.weekly
- ▶ 4. 다음 중 매일, 매주, 매월 실행되는 시스템 작업에 대한 설정을 정의하는 구성 파일은 무엇입니까?
 a. /etc/crontab
 b. /etc/anacrontab
 c. /etc/inittab
 d. /etc/sysconfig/crond
- ▶ 5. 다음 중 정기적으로 임시 파일 정리를 트리거하는 systemd 유닛은 무엇입니까?
 a. systemd-tmpfiles-clean.timer
 b. systemd-tmpfiles-clean.service
 c. dnf-makecache.timer
 d. unbound-anchor.timer

요약

- 반복 실행 사용자 작업은 반복 스케줄에 따라 사용자 작업을 실행합니다.
- 반복 실행 시스템 작업은 반복 스케줄에 따라 시스템 전체에 영향을 주는 관리 작업을 수행합니다.
- `systemd` 타이머 장치는 지연된 작업과 반복 실행 작업을 둘 다 실행할 수 있습니다.

소프트웨어 패키지 설치 및 업데이트

목적

Red Hat 및 DNF 패키지 리포지토리에서 소프트웨어 패키지를 다운로드하고 설치, 업데이트, 관리합니다.

목표

- Red Hat Subscription Management를 사용하여 Red Hat 계정에 시스템을 등록하고 소프트웨어 업데이트 및 지원 서비스 자격을 할당합니다.
- dnf 명령을 사용하여 소프트웨어 패키지를 찾고 설치 및 업데이트합니다.
- 서버의 Red Hat 또는 타사 DNF 리포지토리 사용을 활성화 및 비활성화합니다.

섹션

- Red Hat 지원을 위해 시스템 등록(퀴즈)
- DNF를 사용하여 소프트웨어 패키지 설치 및 업데이트(안내에 따른 연습)
- DNF 소프트웨어 리포지토리 활성화(안내에 따른 연습)

랩

- 소프트웨어 패키지 설치 및 업데이트

Red Hat 지원을 위해 시스템 등록

목표

Red Hat Subscription Management를 사용하여 Red Hat 계정에 시스템을 등록하고 소프트웨어 업데이트 및 지원 서비스 자격을 할당합니다.

Red Hat Subscription Management

Red Hat Subscription Management는 시스템에 제품 서브스크립션 자격을 부여하기 위한 툴을 제공합니다. 관리자는 이 툴을 통해 소프트웨어 패키지 업데이트를 가져오고 시스템이 사용하는 지원 계약 및 서비스 크립션에 대한 정보를 추적할 수 있습니다. `dnf` 명령과 같은 표준 툴은 Red Hat Content Delivery Network에서 제공하는 콘텐츠 배포 네트워크를 통해 소프트웨어 패키지 및 업데이트를 가져옵니다.

Red Hat Subscription Management 툴을 사용하여 다음과 같은 주요 작업을 수행할 수 있습니다.

- 시스템을 등록하여 활성 서브스크립션이 있는 Red Hat 계정과 연결합니다. 서브스크립션 관리자를 사용하면 시스템이 서브스크립션 서비스 인벤토리에 고유하게 등록될 수 있습니다. 사용하지 않을 때는 시스템 등록을 취소할 수 있습니다.
- 시스템을 서브스크립션 하여 선택한 Red Hat 제품의 업데이트 자격을 부여합니다. 서브스크립션은 특정 한 지원 단계, 만료일 및 기본 리포지토리를 가집니다. 이 툴은 특정 자격을 자동 연결하거나 선택하는데 도움이 됩니다.
- 리포지토리 활성화로 소프트웨어 패키지를 제공합니다. 기본적으로 각 서브스크립션이 여러 리포지토리를 활성화합니다. 업데이트 또는 소스 코드와 같은 기타 리포지토리는 활성화되거나 비활성화됩니다. 리포지토리는 소프트웨어 패키지를 저장하고 유지 관리하기 위한 중앙 위치입니다.
- 사용 가능한 자격 또는 사용 중인 자격을 검토 및 추적합니다. Red Hat Customer Portal에서 특정 시스템의 로컬에 있거나 Red Hat 계정과 관련된 서브스크립션 정보를 볼 수 있습니다.

간단한 콘텐츠 액세스

SCA(Simple Content Access)는 Red Hat Subscription Management 기능 중 하나입니다. 조직에서 SCA를 활성화하면 자격 부여 프로세스가 간소화됩니다. SCA를 사용하면 시스템별 수준에서 서브스크립션을 연결할 필요가 없습니다. 시스템을 등록하고 각 시스템에 필요한 리포지토리를 활성화한 다음, 소프트웨어 패키지 설치를 시작합니다.

Simple Content Access(간단한 콘텐츠 액세스)는 Red Hat Satellite Server 및 Red Hat Subscription Management의 선택적 기능입니다. 이 교육 과정에는 SCA를 아직 활성화하지 않은 경우 필요에 따라 서브스크립션 명령이 포함되어 있습니다.

Red Hat 서브스크립션 관리자를 사용하여 시스템 서브스크립션

Red Hat Customer Portal에는 시스템을 등록할 수 있는 다양한 옵션이 있습니다. 예를 들어 GNOME 애플리케이션을 사용하거나 RHEL 웹 콘솔을 통해 그래픽 인터페이스에 액세스하거나 명령줄 툴을 사용하여 시스템을 등록할 수 있습니다.

GNOME 애플리케이션을 사용하여 시스템을 등록하려면 **Activities** 메뉴에서 **Red Hat Subscription Manager** 애플리케이션을 시작합니다. **Type to search** 필드에 `subscription`을 입력하고 **Red Hat Subscription Manager** 애플리케이션을 클릭합니다. 메시지가 표시되면 적절한 암호를 입력하여 인증합니다. **Subscriptions** 창에서 **Register**를 클릭하여 **Register System** 대화 상자를 엽니다.

Register System

URL: Default
 Use proxy server

Method: Account Activation key

Username: <yourusername>

Password:

Organization:

Subscriptions: Attach automatically

Insights: Connect this system to Red Hat Insights.

Register **Cancel**

그림 8.1: 시스템 등록 대화 상자

기본적으로 시스템은 Red Hat Customer Portal에 등록됩니다. Red Hat Customer Portal 계정의 로그인 및 암호를 입력하고 **Register**를 클릭하여 시스템을 등록합니다. 등록되면 시스템이 사용 가능한 서브스크립션을 자동으로 연결합니다.

시스템을 등록하고 서브스크립션에 할당한 후 **Subscriptions** 창을 닫습니다. 이제 시스템이 서브스크립션 되었으며 Red Hat Content Delivery Network에 연결된 서브스크립션에 따라 업데이트를 받거나 새 소프트웨어를 설치할 준비가 되었습니다.

RHEL 웹 콘솔을 사용하여 시스템 서브스크립션

웹 콘솔에 시스템을 등록하려면 권한 있는 사용자로 로그인해야 합니다. **Subscriptions**를 클릭한 다음 **Register**를 클릭합니다.

Search

System

Overview

Logs

Networking

Accounts

Services

Tools

Applications

Diagnostic Reports

Kernel Dump

SELinux

Software Updates !

Subscriptions

Terminal

Overview

Subscription status Not registered

Register

System purpose

Status Unknown

No system purpose attributes set

Installed products

Red Hat Enterprise Linux for x86_64

Auto-attach

Unknown status

그림 8.2: 웹 콘솔 서브스크립션

웹 콘솔의 **Register System** 대화 상자는 GNOME 애플리케이션과 유사합니다. Red Hat Customer Portal 계정의 로그인 및 암호를 입력하고 **Register**를 클릭하여 시스템을 등록합니다. 그런 다음 연결된 서브스크립션에 따라 새 소프트웨어를 설치하거나 시스템을 업데이트할 수 있습니다.

명령줄을 사용하여 시스템 서브스크립션

그래픽 환경을 사용하지 않고 **subscription-manager** 명령을 사용하여 시스템을 등록합니다. **subscription-manager** 명령은 시스템과 가장 호환이 잘 되는 서브스크립션에 시스템을 자동으로 연결합니다.

Red Hat Customer Portal의 자격 증명을 사용하여 **root** 사용자로 시스템을 등록합니다.

```
[root@host ~]# subscription-manager register --username <yourusername>
Registering to: subscription.rhsm.redhat.com:443/subscription
Password: yourpassword
The system has been registered with ID: 1457f7e9-f37e-4e93-960a-c94fe08e1b4f
The registered system name is: host.example.com
```

Red Hat 계정에 사용 가능한 서브스크립션을 봅니다.

```
[root@host ~]# subscription-manager list --available
-----
 Available Subscriptions
-----
 ...output omitted...
```

서브스크립션 자동 첨부:

```
[root@host ~]# subscription-manager attach --auto
...output omitted...
```

또는 사용 가능한 서브스크립션 목록에서 특정 풀의 서브스크립션을 첨부할 수 있습니다.

```
[root@host ~]# subscription-manager attach --pool=poolID
...output omitted...
```

사용된 서브스크립션 보기:

```
[root@host ~]# subscription-manager list --consumed
...output omitted...
```

시스템 등록 취소:

```
[root@host ~]# subscription-manager unregister
Unregistering from: subscription.rhsm.redhat.com:443/subscription
System has been unregistered.
```

활성화 키

활성화 키는 Red Hat Customer Portal을 통해 Red Hat Satellite 서버 및 서브스크립션 관리에서 사용할 수 있는 사전 구성된 서브스크립션 관리 파일입니다. **subscription-manager** 명령에 활성화 키를 사용하

여 사전 정의된 서브스크립션 등록 및 할당을 간소화합니다. 이 등록 방법은 설치 및 배포 자동화에 유용합니다. Simple Content Access(간단한 콘텐츠 액세스)를 활성화하는 조직에서 활성화 키를 사용하면 서브스크립션을 연결하지 않고도 시스템을 등록하고 리포지토리를 활성화할 수 있습니다.

자격 인증서

디지털 인증서는 현재 자격 정보를 로컬 시스템에 저장합니다. 등록된 시스템은 `/etc/pki` 디렉터리에 자격 인증서를 저장합니다.

- `/etc/pki/product` 인증서는 설치된 Red Hat 제품을 나타냅니다.
- `/etc/pki/consumer` 인증서는 등록에 사용되는 Red Hat 계정을 식별합니다.
- `/etc/pki/entitlement` 인증서는 연결된 서브스크립션을 나타냅니다.

`rct` 명령은 인증서를 검사하고, `subscription-manager` 명령은 시스템에서 연결된 서브스크립션을 검사합니다.



참조

`subscription-manager(8)` 및 `rct(8)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/assembling_registering-the-system-and-managing-subscriptions_configuring-basic-system-settings
에서 Registering the System and Managing Subscriptions를 참조하십시오.

▶ 퀴즈

Red Hat 지원을 위해 시스템 등록

다음 질문에 대해 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 사용자 이름 및 암호 없이 Red Hat Subscription Management에 시스템을 등록하는 데 도움이 되는 항목은 무엇입니까?
 - a. 조직 ID
 - b. 프록시 URL
 - c. 활성화 키
 - d. dnf
- ▶ 2. 시스템을 등록하고 서브스크립션하는 데 사용되는 GUI 도구는 무엇입니까?
 - a. PackageKit
 - b. gpk-application
 - c. Red Hat Subscription Manager
 - d. gnome-software
- ▶ 3. 다음 중 자격 인증서를 사용할 때 Red Hat 제품 인증서를 저장하는 디렉터리는 무엇입니까?
 - a. /etc/pki/entitlement
 - b. /etc/subscription/product
 - c. /etc/pki/product
 - d. /etc/certs/pki
 - e. 해당하는 옵션 없음

▶ 솔루션

Red Hat 지원을 위해 시스템 등록

다음 질문에 대해 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 사용자 이름 및 암호 없이 Red Hat Subscription Management에 시스템을 등록하는 데 도움이 되는 항목은 무엇입니까?
 - a. 조직 ID
 - b. 프록시 URL
 - c. 활성화 키
 - d. dnf
- ▶ 2. 시스템을 등록하고 서브스크립션하는 데 사용되는 GUI 도구는 무엇입니까?
 - a. PackageKit
 - b. gpk-application
 - c. Red Hat Subscription Manager
 - d. gnome-software
- ▶ 3. 다음 중 자격 인증서를 사용할 때 Red Hat 제품 인증서를 저장하는 디렉터리는 무엇입니까?
 - a. /etc/pki/entitlement
 - b. /etc/subscription/product
 - c. /etc/pki/product
 - d. /etc/certs/pki
 - e. 해당하는 옵션 없음

DNF를 사용하여 소프트웨어 패키지 설치 및 업데이트

목표

dnf 명령을 사용하여 소프트웨어 패키지를 찾고 설치 및 업데이트합니다.

DNF를 사용하여 소프트웨어 패키지 관리

DNF(Dandified YUM)는 Red Hat Enterprise Linux 9에서 YUM 대신 패키지 관리자로 채택되었습니다. DNF 명령의 기능은 YUM 명령과 동일합니다. 호환성을 위해 YUM 명령도 DNF에 대한 심볼릭 링크로 계속 유지됩니다.

```
[user@host ~]$ ls -l /bin/ | grep yum | awk '{print $9 " " $10 " " $11}'
yum -> dnf-3
yum-builddep -> /usr/libexec/dnf-utils
yum-config-manager -> /usr/libexec/dnf-utils
yum-debug-dump -> /usr/libexec/dnf-utils
yum-debug-restore -> /usr/libexec/dnf-utils
yumdownloader -> /usr/libexec/dnf-utils
yum-groups-manager -> /usr/libexec/dnf-utils
```

이 교육 과정에서는 **dnf** 명령을 사용합니다. 일부 설명서에서 **yum** 명령을 계속 참조할 수도 있지만 파일은 동일한 명령이 연결된 것입니다.

낮은 수준의 **rpm** 명령을 사용하여 패키지를 설치할 수 있지만, 이 명령은 패키지 리포지토리에서 작동하거나 여러 소스의 종속성을 자동으로 해결하도록 설계되어 있지 않습니다.

DNF는 RPM 기반 소프트웨어 설치 및 업데이트를 개선합니다. **dnf** 명령을 사용하면 소프트웨어 패키지 및 해당 종속성을 설치, 업데이트, 제거하고 해당 정보를 가져올 수 있습니다. 트랜잭션 히스토리를 가져오고 여러 개의 Red Hat 및 타사 소프트웨어 리포지토리로 작업할 수 있습니다.

DNF를 사용하여 소프트웨어 찾기

dnf help 명령은 사용법 정보를 표시합니다. **dnf list** 명령은 설치되어 사용 가능한 패키지를 표시합니다.

```
[user@host ~]$ dnf list 'http*'
Available Packages
http-parser.i686          2.9.4-6.el9    rhel-9.0-for-x86_64-appstream-rpms
http-parser.x86_64          2.9.4-6.el9    rhel-9.0-for-x86_64-appstream-rpms
httpcomponents-client.noarch 4.5.13-2.el9  rhel-9.0-for-x86_64-appstream-rpms
httpcomponents-core.noarch   4.4.13-6.el9  rhel-9.0-for-x86_64-appstream-rpms
httpd.x86_64                2.4.51-5.el9  rhel-9.0-for-x86_64-appstream-rpms
httpd-devel.x86_64          2.4.51-5.el9  rhel-9.0-for-x86_64-appstream-rpms
httpd-filesystem.noarch     2.4.51-5.el9  rhel-9.0-for-x86_64-appstream-rpms
httpd-manual.noarch         2.4.51-5.el9  rhel-9.0-for-x86_64-appstream-rpms
httpd-tools.x86_64          2.4.51-5.el9  rhel-9.0-for-x86_64-appstream-rpms
```

dnf search KEYWORD 명령은 이름 및 요약 필드에 있는 키워드만으로 패키지를 표시합니다. 이름, 요약 및 설명 필드에 "web server"가 포함된 패키지를 검색하려면 **search all**을 사용합니다.

```
[user@host ~]$ dnf search all 'web server'
=====
Summary & Description Matched: web server =====
nginx.x86_64 : A high performance web server and reverse proxy server
pcp-pmda-weblog.x86_64 : Performance Co-Pilot (PCP) metrics from web server logs
=====
Summary Matched: web server =====
libcurl.x86_64 : A library for getting files from web servers
libcurl.i686 : A library for getting files from web servers
=====
Description Matched: web server =====
freeradius.x86_64 : High-performance and highly configurable free RADIUS server
git-instaweb.noarch : Repository browser in gitweb
http-parser.i686 : HTTP request/response parser for C
http-parser.x86_64 : HTTP request/response parser for C
httpd.x86_64 : Apache HTTP Server
mod_auth_openidc.x86_64 : OpenID Connect auth module for Apache HTTP Server
mod_jk.x86_64 : Tomcat mod_jk connector for Apache
mod_security.x86_64 : Security module for the Apache HTTP Server
varnish.i686 : High-performance HTTP accelerator
varnish.x86_64 : High-performance HTTP accelerator
...output omitted...
```

dnf info PACKAGE NAME 명령은 설치에 필요한 디스크 공간을 포함하여 자세한 패키지 정보를 반환합니다. 예를 들어 다음 명령은 **httpd** 패키지에 대한 정보를 검색합니다.

```
[user@host ~]$ dnf info httpd
Available Packages
Name        : httpd
Version     : 2.4.51
Release     : 5.el9
Architecture: x86_64
Size        : 1.5 M
Source      : httpd-2.4.51-5.el9.src.rpm
Repository   : rhel-9.0-for-x86_64-appstream-rpms
Summary     : Apache HTTP Server
URL         : https://httpd.apache.org/
License      : ASL 2.0
Description  : The Apache HTTP Server is a powerful, efficient, and extensible
               : web server.
```

dnf provides PATHNAME 명령은 지정된 경로 이름(경로 이름에 와일드카드 문자가 포함된 경우가 많음)과 일치하는 패키지를 표시합니다. 예를 들어 다음 명령은 **/var/www/html** 디렉터리를 제공하는 패키지를 찾습니다.

```
[user@host ~]$ dnf provides /var/www/html
httpd-filesystem-2.4.51-5.el9.noarch : The basic directory layout for the Apache
                                         HTTP Server
Repo       : rhel-9.0-for-x86_64-appstream-rpms
Matched from:
Filename   : /var/www/html
```

DNF를 사용하여 소프트웨어 설치 및 제거

`dnf install PACKAGE_NAME` 명령은 종속성을 포함하여 소프트웨어 패키지를 가져와서 설치합니다.

```
[root@host ~]# dnf install httpd
Dependencies resolved.
=====
 Package      Arch    Version       Repository      Size
=====
Installing:
 httpd        x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 1.5 M
Installing dependencies:
 apr          x86_64  1.7.0-11.el9   rhel-9.0-for-x86_64-appstream-rpms 127 k
 apr-util     x86_64  1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 98 k
 apr-util-bdb x86_64  1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 15 k
 httpd-filesystem noarch 2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 17 k
 httpd-tools   x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 88 k
 redhat-logos-httpd
                  noarch 90.4-1.el9     rhel-9.0-for-x86_64-appstream-rpms 18 k
Installing weak dependencies:
 apr-util-openssl x86_64  1.6.1-20.el9   rhel-9.0-for-x86_64-appstream-rpms 17 k
 mod_http2       x86_64  1.15.19-2.el9   rhel-9.0-for-x86_64-appstream-rpms 153 k
 mod_lua         x86_64  2.4.51-5.el9   rhel-9.0-for-x86_64-appstream-rpms 63 k

Transaction Summary
=====
Install 10 Packages

Total download size: 2.1 M
Installed size: 5.9 M
Is this ok [y/N]: y
Downloading Packages:
(1/10): apr-1.7.0-11.el9.x86_64.rpm           6.4 MB/s | 127 kB     00:00
(2/10): apr-util-bdb-1.6.1-20.el9.x86_64.rpm   625 kB/s |  15 kB     00:00
(3/10): apr-util-openssl-1.6.1-20.el9.x86_64.r p 1.9 MB/s |  17 kB     00:00
...output omitted...
Total                                         24 MB/s | 2.1 MB     00:00

Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
Preparing      :                                                 1/1
Installing    : apr-1.7.0-11.el9.x86_64             1/10
Installing    : apr-util-bdb-1.6.1-20.el9.x86_64   2/10
Installing    : apr-util-openssl-1.6.1-20.el9.x86_64 3/10
...output omitted...
Installed:
  apr-1.7.0-11.el9.x86_64           apr-util-1.6.1-20.el9.x86_64
  apr-util-bdb-1.6.1-20.el9.x86_64  apr-util-openssl-1.6.1-20.el9.x86_64
...output omitted...
Complete!
```

8장 | 소프트웨어 패키지 설치 및 업데이트

dnf update PACKAGE NAME 명령은 종속성을 포함하여 지정된 패키지의 이후 버전을 가져와서 설치합니다. 일반적으로 이 프로세스는 구성 파일을 제자리에 보존하려고 하지만, 업데이트 후에 이전 이름이 작동하지 않을 것이라고 패키지 작성자가 생각할 경우 파일 이름이 변경될 수도 있습니다. PACKAGE NAME을 지정하지 않으면 관련 업데이트가 모두 설치됩니다.

```
[root@host ~]# dnf update
```

새 커널로 부팅하는 방법으로만 커널을 테스트할 수 있기 때문에 패키지는 여러 버전을 한 번에 설치할 수 있도록 구체적으로 지원합니다. 새 커널이 부팅되지 않는 경우 이전 커널을 계속 사용할 수 있습니다. **dnf update kernel** 명령을 실행하면 새 커널이 설치됩니다. 구성 파일에는 관리자가 업데이트를 요구하는 경우에도 항상 설치할 패키지 목록이 있습니다.

dnf list kernel 명령으로 설치되어 있고 사용 가능한 커널을 모두 표시합니다. 현재 실행 중인 커널을 보려면 **uname** 명령을 사용합니다. **uname** 명령의 **-r** 옵션은 커널 버전과 릴리스만 표시하고, **uname** 명령의 **-a** 옵션은 커널 릴리스와 추가 정보를 표시합니다.

```
[user@host ~]$ dnf list kernel
Installed Packages
kernel.x86_64                  5.14.0-70.el9          @System
[user@host ~]$ uname -r
5.14.0-70.el9.x86_64
[user@host ~]$ uname -a
Linux workstation.lab.example.com 5.14.0-70.el9.x86_64 #1 SMP PREEMPT Thu Feb 24
19:11:22 EST 2022 x86_64 x86_64 x86_64 GNU/Linux
```

dnf remove PACKAGE NAME 명령은 지원되는 패키지를 포함하여 설치된 소프트웨어 패키지를 제거합니다.

```
[root@host ~]# dnf remove httpd
```

**경고**

dnf remove 명령은 표시된 패키지와 해당 패키지 제거를 요구하는 모든 패키지 (및 해당 패키지가 필요한 패키지 등)를 제거합니다. 이 명령으로 인해 패키지가 예기치 않게 제거될 수 있으므로 제거되는 패키지 목록을 검토합니다.

DNF를 사용하여 소프트웨어 그룹 설치 및 제거

dnf 명령에는 함께 설치된 관련 소프트웨어 컬렉션인 그룹 개념도 있습니다.

Red Hat Enterprise Linux 9에서 **dnf** 명령은 두 종류의 패키지 그룹을 설치할 수 있습니다. 정규 그룹은 패키지 컬렉션입니다. 환경 그룹은 정규 그룹의 컬렉션입니다. 컬렉션에서 제공하는 패키지 또는 그룹은 **mandatory**(그룹이 설치된 경우 설치해야 함), **default**(그룹이 설치된 경우 일반적으로 설치됨) 또는 **optional**(구체적으로 요청되지 않은 한, 그룹을 설치할 때 설치되지 않음)로 표시될 수 있습니다.

dnf list 명령과 유사하게, **dnf group list** 명령은 설치되어 사용 가능한 그룹의 이름을 표시합니다.

```
[user@host ~]$ dnf group list
Available Environment Groups:
  Server with GUI
  Server
  Minimal Install
```

```
...output omitted...
Available Groups:
  Legacy UNIX Compatibility
  Console Internet Tools
  Container Management
...output omitted...
```

일부 그룹은 환경 그룹을 통해 설치되며 기본적으로 숨겨집니다. 이러한 숨겨진 그룹은 **dnf group list hidden** 명령을 사용하여 표시합니다.

dnf group info 명령은 그룹에 대한 정보를 표시합니다. 여기에는 필수, 기본 및 선택 사항 패키지 이름 목록이 포함됩니다.

```
[user@host ~]$ dnf group info "RPM Development Tools"
Group: RPM Development Tools
Description: Tools used for building RPMs, such as rpmbuild.
Mandatory Packages:
  redhat-rpm-config
  rpm-build
Default Packages:
  rpmdevtools
Optional Packages:
  rpmlint
```

dnf group install 명령은 필수 및 기본 패키지와 해당 종속 패키지를 설치하는 그룹을 설치합니다.

```
[root@host ~]# dnf group install "RPM Development Tools"
...output omitted...
Installing Groups:
  RPM Development Tools

Transaction Summary
=====
Install 19 Packages

Total download size: 4.7 M
Installed size: 15 M
Is this ok [y/N]: y
...output omitted...
```

**중요**

Red Hat Enterprise Linux 7부터 Yum 그룹의 동작이 변경되어 오브젝트로 취급되고 시스템에서 추적됩니다. 설치된 그룹이 업데이트되었으며 Yum 리포지토리가 새로운 필수 또는 기본 패키지를 그룹에 추가한 경우 업데이트할 때 새 패키지가 설치됩니다.

RHEL 6 및 이전 버전에서는 모든 필수 패키지가 설치되었거나, 필수 패키지가 없거나, 그룹의 기본 또는 선택적 패키지가 설치된 경우 그룹이 설치되는 것으로 간주합니다. RHEL 7부터 **yum group install**을 사용하여 설치한 경우에만 그룹이 설치되는 것으로 간주됩니다. **yum group mark install GROUPNAME** 명령을 사용하여 그룹을 설치된 것으로 표시할 수 있으며, 누락된 패키지와 해당 종속성은 다음 업데이트 시 설치됩니다.

RHEL 6 및 이전 버전에는 두 단어로 된 **yum group** 명령이 없었습니다. 즉, RHEL 6에는 **yum grouplist** 명령이 있었지만 해당하는 RHEL 7 및 RHEL 8 **yum group list** 명령은 없었습니다.

트랜잭션 히스토리 보기

모든 설치 및 제거 트랜잭션은 `/var/log/dnf.rpm.log` 파일에 기록됩니다.

```
[user@host ~]$ tail -5 /var/log/dnf.rpm.log
2022-03-23T16:46:43-0400 SUBDEBUG Installed: python-srpm-macros-3.9-52.el9.noarch
2022-03-23T16:46:43-0400 SUBDEBUG Installed: redhat-rpm-config-194-1.el9.noarch
2022-03-23T16:46:44-0400 SUBDEBUG Installed: elfutils-0.186-1.el9.x86_64
2022-03-23T16:46:44-0400 SUBDEBUG Installed: rpm-build-4.16.1.3-11.el9.x86_64
2022-03-23T16:46:44-0400 SUBDEBUG Installed: rpmdevtools-9.5-1.el9.noarch
```

dnf history 명령은 설치 및 제거 트랜잭션에 대한 요약을 표시합니다.

ID	Command line	Date and time	Action(s)	Altered
7	group install RPM Develop	2022-03-23 16:46	Install	20
6	install httpd	2022-03-23 16:21	Install	10 EE
5	history undo 4	2022-03-23 15:04	Removed	20
4	group install RPM Develop	2022-03-23 15:03	Install	20
3		2022-03-04 03:36	Install	5
2		2022-03-04 03:33	Install	767 EE
1	-y install patch ansible-	2022-03-04 03:31	Install	80

dnf history undo 명령은 트랜잭션을 되돌립니다.

```
[root@host ~]# dnf history undo 6
...output omitted...
Removing:
  apr-util-openssl x86_64 1.6.1-20.el9 @rhel-9.0-for-x86_64-appstream-rpms 24 k
  httpd           x86_64 2.4.51-5.el9 @rhel-9.0-for-x86_64-appstream-rpms 4.7 M
...output omitted...
```

DNF 명령 요약

이름 또는 패키지 그룹별로 패키지를 찾고 설치, 업데이트 및 제거할 수 있습니다.

작업:	명령:
설치되어 사용 가능한 패키지를 이름으로 나열합니다.	<code>dnf list [NAME-PATTERN]</code>
설치되어 사용 가능한 그룹을 나열합니다.	<code>dnf group list</code>
패키지를 키워드로 검색합니다.	<code>dnf search KEYWORD</code>
패키지 세부 정보를 표시합니다.	<code>dnf info PACKAGE_NAME</code>
패키지를 설치합니다.	<code>dnf install PACKAGE_NAME</code>
패키지 그룹을 설치합니다.	<code>dnf group install GROUPNAME</code>
모든 패키지를 업데이트합니다.	<code>dnf update</code>
패키지를 제거합니다.	<code>dnf remove PACKAGE_NAME</code>
트랜잭션 내역을 표시합니다.	<code>dnf history</code>

DNF를 사용하여 패키지 모듈 스트림 관리

일반적으로 애플리케이션 소프트웨어 패키지 및 관련 패키지의 대체 버전을 관리하려면 버전마다 다른 리포지토리를 유지 관리해야 합니다. 최신 버전의 애플리케이션을 원하는 개발자와 가장 안정적인 버전의 애플리케이션을 원하는 관리자에게 이 상황은 관리하기에 번거로운 것이었습니다. Red Hat은 모듈성이라는 기술을 사용하여 이 프로세스를 간소화합니다. 모듈성을 사용하면 단일 리포지토리에서 여러 버전의 애플리케이션 패키지와 해당 종속성을 호스팅할 수 있습니다.

BaseOS 및 애플리케이션 스트림 소개

Red Hat Enterprise Linux 9에서는 두 개의 주요 소프트웨어 리포지토리인 BaseOS 및 애플리케이션 스트림(AppStream)을 통해 콘텐츠를 배포합니다.

BaseOS 리포지토리는 Red Hat Enterprise Linux의 핵심 운영 체제 콘텐츠를 RPM 패키지로 제공합니다. BaseOS 구성 요소의 라이프사이클은 이전 Red Hat Enterprise Linux 릴리스의 콘텐츠와 동일합니다. 애플리케이션 스트림 리포지토리는 다양한 라이프사이클의 콘텐츠를 모듈 및 기존 패키지로 모두 제공합니다.

애플리케이션 스트림에는 시스템의 필수 부분은 물론 Red Hat Software Collections 및 기타 제품과 프로그램의 일부로 이전부터 사용할 수 있었던 다양한 애플리케이션이 포함됩니다. 각 애플리케이션 스트림의 라이프사이클은 Red Hat Enterprise Linux 9와 동일하거나 더 짧습니다.

BaseOS와 AppStream은 모두 Red Hat Enterprise Linux 9 시스템의 필수 부분입니다.

애플리케이션 스트림 리포지토리에는 모듈과 기존 RPM 패키지라는 두 가지 유형의 콘텐츠가 포함되어 있습니다. 모듈은 함께 포함된 일련의 RPM 패키지를 나타냅니다. 모듈은 다양한 버전의 애플리케이션을 설치할 수 있도록 여러 스트림을 포함할 수 있습니다. 모듈 스트림을 사용하면 시스템에 해당 모듈 스트림 내의 RPM 패키지에 액세스할 수 있는 액세스 권한이 부여됩니다. 일반적으로 모듈은 특정 버전의 소프트웨어 애플리케이션 또는 프로그래밍 언어를 중심으로 RPM 패키지를 구성합니다. 일반적인 모듈에는 애플리케이션이 있는 패키지, 애플리케이션의 특정 종속성 라이브러리가 있는 패키지, 애플리케이션 설명서가 있는 패키지, 도우미 유틸리티가 있는 패키지가 포함됩니다.

**중요**

Red Hat Enterprise Linux 9.0은 모듈 없이 제공됩니다. RHEL 9의 향후 버전에서는 추가 콘텐츠 및 이후 소프트웨어 버전이 모듈로 도입될 수 있습니다. 또한 RHEL 9부터 기본 모듈 스트림이 더 이상 기본적으로 정의되지 않으므로 수동으로 지정해야 합니다. `/etc/dnf/modules.defaults.d/` 디렉터리의 구성 파일을 사용하여 기본 모듈 스트림을 정의할 수 있습니다.

모듈 스트림

각 모듈에는 각기 다른 버전의 콘텐츠가 포함된 하나 이상의 모듈 스트림이 있습니다. 각 스트림은 독립적으로 업데이트를 받습니다. 모듈 스트림은 애플리케이션 스트림의 실제 리포지토리에 있는 가상 저장소로 간주 할 수 있습니다.

각 모듈에 대해 해당 스트림 중 하나만 활성화할 수 있으며, 이 스트림이 패키지를 제공합니다.

모듈 프로필

각 모듈에는 프로필이 1개 이상 있을 수 있습니다. 프로필은 서버, 클라이언트, 개발, 최소 설치 등의 특정 사용 사례를 위해 함께 설치할 수 있는 패키지 목록입니다.

모듈 프로필을 설치하면 모듈 스트림에서 특정 패키지 세트가 설치됩니다. 이후에 정상적으로 패키지를 설치하거나 제거할 수 있습니다. 프로필을 지정하지 않으면 모듈은 해당 기본 프로필을 설치합니다.

DNF를 사용하여 모듈 관리

Red Hat Enterprise Linux 9에서는 애플리케이션 스트림의 모듈식 기능을 지원합니다. 모듈식 콘텐츠를 처리하려면 `dnf module` 명령을 사용할 수 있습니다. 그러지 않으면 `dnf` 명령은 일반 패키지와 유사하게 모듈에서 작동합니다.

모듈을 관리하기 위한 중요한 명령은 다음 목록을 참조하십시오.

- **`dnf module list`** : 모듈 이름, 스트림, 프로필 및 요약과 함께 사용 가능한 모듈을 표시합니다.
- **`dnf module list module-name`**: 특정 모듈에 대한 모듈 스트림을 표시하고 해당 상태를 검색합니다.
- **`dnf module info module-name`** : 사용 가능한 프로필 및 모듈이 설치하는 패키지 목록을 포함하여 모듈의 세부 정보를 표시합니다. 모듈 스트림을 지정하지 않고 `dnf module info` 명령을 실행하면 기본 프로필 및 스트림에서 설치된 패키지가 표시됩니다. 특정 모듈 스트림을 보려면 `module-name:stream` 형식을 사용하십시오. 각 모듈 프로필에서 설치한 패키지에 대한 정보를 표시하려면 `--profile` 옵션을 추가 합니다.
- **`dnf module provides package`** : 특정 패키지를 제공하는 모듈을 표시합니다.



참조

`dnf(1)` 및 `dnf.conf(5)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-software-packages_configuring-basic-system-settings

에서 Red Hat Enterprise Linux 9 Configuring Basic system Settings Guide의 Managing Software Packages 장을 참조하십시오.

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_software_with_the_dnf_tool/index#assembly_distribution-of-content-in-rhel-9_managing-software-with-the-dnf-tool

에서 Red Hat Enterprise Linux 9 Managing Software with the DNF Tool Guide의 Distribution of Content in RHEL 9 장을 참조하십시오.

모듈성

<https://docs.fedoraproject.org/en-US/modularity/>

▶ 연습 가이드

DNF를 사용하여 소프트웨어 패키지 설치 및 업데이트

이 연습에서는 패키지 및 패키지 그룹을 설치하고 제거합니다.

결과

- 종속성과 함께 패키지를 설치하고 제거합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start software-dnf
```

지침

- ▶ 1. **workstation**에서 **student** 사용자로 **servera** 시스템에 대한 SSH 세션을 엽니다. **sudo -i** 명령을 사용하여 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 특정 패키지를 검색합니다.

- 2.1. **nmap** 명령 실행을 시도해 봅니다. 패키지가 설치되지 않았다는 것을 확인해야 합니다.

```
[root@servera ~]# nmap
-bash: nmap: command not found
```

- 2.2. **dnf search** 명령을 사용하여 이름 또는 요약에 **nmap**이 포함된 패키지를 검색합니다.

```
[root@servera ~]# dnf search nmap
...output omitted...
===== Name Exactly Matched: nmap =====
nmap.x86_64 : Network exploration tool and security scanner
===== Name & Summary Matched: nmap =====
nmap-ncat.x86_64 : Nmap's Netcat replacement
```

- 2.3. **dnf info** 명령을 사용하여 **nmap** 패키지에 대한 자세한 정보를 얻습니다.

```
[root@servera ~]# dnf info nmap
...output omitted...
Available Packages
Name        : nmap
Epoch       : 3
Version     : 7.91
Release    : 10.el9
...output omitted...
```

▶ 3. `dnf install` 명령을 사용하여 `nmap` 패키지를 설치합니다.

```
[root@servera ~]# dnf install nmap
...output omitted...
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Installing:
  nmap           x86_64   3:7.91-10.el9   rhel-9.0-for-x86_64-appstream-rpms  5.6 M

Transaction Summary
=====
Install 1 Package

Total download size: 5.6 M
Installed size: 24 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

▶ 4. 패키지를 제거합니다.

- 4.1. `dnf remove` 명령을 사용하여 `nmap` 패키지를 제거합니다. 그러나 메시지가 표시되면 `no`로 응답합니다. 제거되는 패키지는 몇 개입니까?

```
[root@servera ~]# dnf remove nmap
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Removing:
  nmap           x86_64   3:7.91-10.el9   @rhel-9.0-for-x86_64-appstream-rpms  24 M

Transaction Summary
=====
Remove 1 Package
```

```
Freed space: 24 M
Is this ok [y/N]: n
Operation aborted.
```

- 4.2. **dnf remove** 명령을 사용하여 **tar** 패키지를 제거합니다. 그러나 메시지가 표시되면 **no**로 응답합니다. 제거되는 패키지는 몇 개입니까?

```
[root@servera ~]# dnf remove tar
...output omitted...
Dependencies resolved.

=====
Package      Arch    Version       Repository      Size
=====
Removing:
tar          x86_64  2:1.34-3.el9   @System           3.0 M
Removing dependent packages:
cockpit      x86_64  264-1.el9     @rhel-9.1-for-x86_64-baseos-rpms  57 k
cockpit-system noarch 264-1.el9     @System           3.3 M
...output omitted...

Transaction Summary
=====
Remove 12 Packages

Freed space: 48 M
Is this ok [y/N]: n
Operation aborted.
```

- ▶ 5. "보안 툴" 구성 요소 그룹에 대한 정보를 수집하고 **servera**에 설치합니다.

- 5.1. **dnf group list** 명령으로 사용 가능한 모든 구성 요소 그룹을 표시합니다.

```
[root@servera ~]# dnf group list
```

- 5.2. **dnf group info** 명령을 사용하여 포함된 패키지 목록을 비롯하여 **Security Tools** 구성 요소 그룹에 대한 자세한 정보를 얻습니다.

```
[root@servera ~]# dnf group info "Security Tools"
...output omitted...
Group: Security Tools
Description: Security tools for integrity and trust verification.
Default Packages:
scap-security-guide
Optional Packages:
aide
hmaccalc
openscap
openscap-engine-sce
openscap-utils
scap-security-guide-doc
scap-workbench
```

```
tpm2-tools
tss2
udica
```

5.3. `dnf group install` 명령을 사용하여 **Security Tools** 구성 요소 그룹을 설치합니다.

```
[root@servera ~]# dnf group install "Security Tools"
...output omitted...
Dependencies resolved.
=====
Package      Arch    Version       Repository      Size
=====
Installing group/module packages:
scap-security-guide
noarch 0.1.60-5.el9  rhel-9.0-for-x86_64-appstream-rpms 683 k
Installing dependencies:
openscap      x86_64  1:1.3.6-3.el9  rhel-9.0-for-x86_64-appstream-rpms 2.0 M
...output omitted...

Transaction Summary
=====
Install 5 Packages

Total download size: 3.0 M
Installed size: 94 M
Is this ok [y/N]: y
...output omitted...
Installed:
  openscap-1:1.3.6-3.el9.x86_64
  openscap-scanner-1:1.3.6-3.el9.x86_64
  scap-security-guide-0.1.60-5.el9.noarch
  xmlsec1-1.2.29-9.el9.x86_64
  xmlsec1-openssl-1.2.29-9.el9.x86_64

Complete!
```

▶ 6. `dnf` 명령의 히스토리 및 실행 취소 옵션을 살펴봅니다.

6.1. `dnf history` 명령을 사용하여 최근의 `dnf` 기록을 표시합니다.

```
[root@servera ~]# dnf history
ID      | Command line           | Date and time   | Action(s)      | Altered
-----
3 | group install Security T | 2022-03-24 15:23 | Install        | 6
2 | install nmap            | 2022-03-24 15:12 | Install        | 1
1 | -y install @base firewall | 2022-03-03 04:47 | Install        | 156 EE
```

사용자 시스템의 기록은 다를 수 있습니다.

6.2. `dnf history info` 명령을 사용하여 마지막 트랜잭션이 그룹 설치인지 확인합니다. 다음 명령에서 트랜잭션 ID를 이전 단계의 ID로 교체합니다.

```
[root@servera ~]# dnf history info 3
Transaction ID : 3
Begin time      : Thu 24 Mar 2022 03:23:56 PM EDT
Begin rpmdb     : 7743aed72ac79f632442c9028aafdf2499a1591f92a660b3f09219b422ca95f02
End time        : Thu 24 Mar 2022 03:23:58 PM EDT (2 seconds)
End rpmdb       : 20c4f021538b7dca9a874260784b1e5cf9bc142da869967269e3d84dd0f789d
User           : Student User <student>
Return-Code     : Success
Releasever     : 9
Command Line   : group install Security Tools
Comment        :
Packages Altered:
    Install openscap-1:1.3.6-3.el9.x86_64          @rhel-9.0-for-x86_64-
appstream-rpms
    Install openscap-scanner-1:1.3.6-3.el9.x86_64  @rhel-9.0-for-x86_64-
appstream-rpms
...output omitted...
```

- 6.3. `dnf history undo` 명령을 사용하여 `nmap` 패키지를 설치할 때 설치된 패키지 집합을 제거합니다. 시스템의 `dnf history` 명령 출력에서 올바른 트랜잭션 ID를 찾은 후 다음 명령에 서 해당 ID를 사용합니다.

```
[root@servera ~]# dnf history undo 2
```

▶ 7. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish software-dnf
```

이것으로 섹션을 완료합니다.

DNF 소프트웨어 리포지토리 활성화

목표

서버의 Red Hat 또는 타사 DNF 리포지토리 사용을 활성화 및 비활성화합니다.

Red Hat 소프트웨어 리포지토리 활성화

시스템은 수많은 Red Hat 리포지토리에 액세스할 수 있는 경우가 많습니다. `dnf repolist all` 명령은 사용 가능한 모든 리포지토리와 해당 상태를 표시합니다.

```
[user@host ~]$ dnf repolist all
repo id                                repo name          status
rhel-9.0-for-x86_64-appstream-rpms      RHEL 9.0 AppStream    enabled
rhel-9.0-for-x86_64-baseos-rpms         RHEL 9.0 BaseOS      enabled
```



참고

Red Hat 서브스크립션은 특정 리포지토리에 대한 액세스 권한을 부여합니다. 과거에는 관리자가 시스템별로 서브스크립션을 연결해야 했습니다. SCA(Simple Content Access)는 시스템이 리포지토리에 액세스하는 방법을 간소화합니다. SCA를 사용하면 시스템이 서브스크립션을 연결하지 않고도 구매한 모든 서브스크립션에서 임의 리포지토리에 액세스할 수 있습니다. Red Hat Customer Portal의 [My Subscriptions > Subscription Allocations](#)에서 또는 Red Hat Satellite 서버에서 SCA를 활성화할 수 있습니다.

`dnf config-manager` 명령은 리포지토리를 활성화 및 비활성화할 수 있습니다. 예를 들어, 다음 명령을 실행하면 `rhel-9-server-debug-rpms` 리포지토리가 활성화됩니다.

```
[user@host ~]$ dnf config-manager --enable rhel-9-server-debug-rpms
```

비 Red Hat 소스는 타사 리포지토리를 통해 소프트웨어를 제공합니다. 예를 들어 Adobe는 DNF 리포지토리를 통해 Linux용 소프트웨어 일부를 제공합니다. Red Hat 강의실에서 `content.example.com` 서버는 DNF 리포지토리를 호스팅합니다. `dnf` 명령은 웹 사이트, FTP 서버 또는 로컬 파일 시스템에서 리포지토리에 액세스할 수 있습니다.

두 가지 방법 중 하나로 타사 리포지토리를 추가할 수 있습니다. `/etc/yum.repos.d/` 디렉터리에 `.repo` 파일을 생성하거나 `/etc/dnf/dnf.conf` 파일에 `[repository]` 섹션을 추가할 수 있습니다. `.repo` 파일을 사용하고 추가 리포지토리 구성은 `dnf.conf` 파일을 예약하는 것이 좋습니다. `dnf` 명령은 기본적으로 두 위치를 모두 검색하지만 `.repo` 파일이 우선합니다. `.repo` 파일에는 리포지토리 URL, 이름, GPG를 사용하여 패키지 서명을 확인할지 여부, 확인하는 경우 신뢰할 수 있는 GPG 키를 가리키는 URL이 포함되어 있습니다.

DNF 리포지토리 추가

`dnf config-manager` 명령은 시스템에 리포지토리를 추가할 수도 있습니다. 다음 명령은 기존 리포지토리의 URL을 사용하여 `.repo` 파일을 생성합니다.

```
[user@host ~]$ dnf config-manager \
--add-repo="https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/"
Adding repo from: https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
```

해당 .repo 파일은 /etc/yum.repos.d/ 디렉터리에 표시됩니다.

```
[user@host ~]$ cd /etc/yum.repos.d
[user@host yum.repos.d]$ cat \
dl.fedoraproject.org_pub_epel_9_Everything_x86_64_.repo
[dl.fedoraproject.org_pub_epel_9_Everything_x86_64_]
name=created by dnf config-manager from https://dl.fedoraproject.org/pub/epel/9/
Everything/x86_64/
baseurl=https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
enabled=1
```

rpm 명령은 GPG 키를 사용하여 패키지에 서명하고, 공개 키를 가져와 패키지의 무결성과 신뢰성을 확인합니다. **dnf** 명령은 리포지토리 구성 파일을 사용하여 GPG 공개 키 위치를 제공하고 키를 가져와서 패키지를 확인합니다. 키는 원격 리포지토리 사이트의 다양한 위치(예: <http://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-9>)에 저장됩니다. 관리자는 **dnf** 명령을 통해 외부 소스에서 키를 검색하는 대신 키를 로컬 파일로 다운로드해야 합니다. 예를 들어 다음 .repo 파일은 **gpgkey** 매개 변수를 사용하여 로컬 키를 참조합니다.

```
[EPEL]
name=EPEL 9
baseurl=https://dl.fedoraproject.org/pub/epel/9/Everything/x86_64/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-9
```

로컬 리포지토리용 RPM 구성 패키지

일부 리포지토리는 구성 파일과 GPG 공개 키를 RPM 패키지의 일부로 제공하여 설치를 간소화합니다. **rpm --import** 명령을 사용하여 GPG 공개 키를 가져올 수 있습니다. **dnf install** 명령은 이러한 RPM 패키지를 다운로드하고 설치할 수 있습니다.

예를 들어 다음 명령은 **RPM-GPG-KEY-EPEL-9**(EPEL) GPG 공개 키를 가져와서 RHEL9 EPEL(Extra Packages for Enterprise Linux) 리포지토리 RPM을 설치합니다.

```
[user@host ~]$ rpm --import \
https://dl.fedoraproject.org/pub/epel/RPM-GPG-KEY-EPEL-9
[user@host ~]$ dnf install \
https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```



경고

서명된 패키지를 설치하기 전에 RPM GPG 키를 가져와서 패키지가 신뢰할 수 있는 소스에서 가져온 것인지 확인합니다. RPM GPG 키를 가져오지 않은 경우 **dnf** 명령은 서명된 패키지를 설치하지 못합니다.

dnf 명령의 **--nogpgcheck** 옵션은 누락된 GPG 키를 무시하지만 손상되거나 위조된 패키지가 설치될 수 있습니다.

.repo 파일은 여러 개의 리포지토리 참조를 파일 하나에 표시하는 경우가 많습니다. 각 리포지토리에 대한 참조는 팔호 안에 든 한 단어의 이름으로 시작됩니다.

```
[user@host ~]$ cat /etc/yum.repos.d/epel.repo
[epel]
name=Extra Packages for Enterprise Linux $releasever - $basearch
#baseurl=https://download.example/pub/epel/$releasever/Everything/$basearch/
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-$releasever&arch=
$basearch&infra=$infra&content=$contentdir
enabled=1
gpgcheck=1
countme=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever
...output omitted...
[epel-source]
name=Extra Packages for Enterprise Linux $releasever - Source
#baseurl=https://download.example/pub/epel/$releasever/Everything/source/tree/
metalink=https://mirrors.fedoraproject.org/metalink?repo=epel-source-
$releasever&arch=$basearch&infra=$infra&content=$contentdir
enabled=0
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-$releasever
gpgcheck=1
```

리포지토리를 정의하지만 기본적으로 검색하지 않으려면 `enabled=0` 매개 변수를 삽입합니다. `dnf config-manager` 명령은 리포지토리를 영구적으로 활성화 및 비활성화하지만 `dnf` 명령의 `--enablerepo=PATTERN` 및 `--disablerepo=PATTERN` 옵션은 명령이 실행되는 동안 일시적으로 리포지토리를 활성화 및 비활성화합니다.



참조

`dnf(8)`, `dnf.conf(5)` 및 `dnf-config-manager(8)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_software_with_the_dnf_tool

에서 Red Hat Enterprise Linux 9 제품 설명서의 Managing Software with the DNF Tool 장을 참조하십시오.

▶ 연습 가이드

DNF 소프트웨어 리포지토리 활성화

이 연습에서는 원격 DNF 리포지토리에서 패키지를 가져온 다음, 해당 리포지토리에서 패키지를 업데이트하거나 설치하도록 서버를 구성합니다.

결과

- 강의실 서버에서 소프트웨어 업데이트를 가져오도록 시스템을 구성하고, 최신 패키지를 사용하도록 시스템을 업데이트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start software-repo
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다. **sudo -i** 명령을 사용하여 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 다음 URL에서 사용자 지정 패키지 및 업데이트를 가져오도록 **servera**의 소프트웨어 리포지토리를 구성합니다.

- http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht**의 사용자 지정 패키지
- http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata**의 사용자 지정 패키지 업데이트

- 2.1. **dnf config-manager** 명령을 사용하여 사용자 지정 리포지토리를 추가합니다.

```
[root@servera ~]# dnf config-manager \
--add-repo "http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht"
Adding repo from: http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht
```

- 2.2. 이전 명령이 **/etc/yum.repos.d** 디렉터리에 생성한 소프트웨어 리포지토리 파일을 검사합니다. **vim** 명령을 사용하여 파일을 편집하고 **gpgcheck=0** 매개 변수를 추가하여 리포지토리에 대한 GPG 키 확인을 비활성화합니다.

```
[root@servera ~]# vim \
/etc/yum.repos.d/content.example.com_rhel9.0_x86_64_rhcsa-practice_rht.repo
[content.example.com_rhel9.0_x86_64_rhcsa-practice_rht]
name=created by dnf config-manager from http://content.example.com/rhel9.0/x86_64/
rhcsa-practice/rht
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/rht
enabled=1
gpgcheck=0
```

- 2.3. `/etc/yum.repos.d/errata.repo` 파일을 만들어 다음 콘텐츠를 포함하는 업데이트 리포지토리를 활성화합니다.

```
[rht-updates]
name=rht updates
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

- 2.4. `dnf repolist all` 명령을 사용하여 시스템의 모든 리포지토리를 표시합니다.

```
[root@servera ~]# dnf repolist all
repo id                                repo name      status
content.example.com_rhel9.0_x86_64_rhcsa-practice_rht created by .... enabled
...output omitted...
rht-updates                               rht updates    enabled
```

▶ 3. `rht-updates` 소프트웨어 리포지토리를 비활성화하고 `rht-system` 패키지를 설치합니다.

- 3.1. `dnf config-manager --disable` 명령을 사용하여 `rht-updates` 리포지토리를 비활성화합니다.

```
[root@servera ~]# dnf config-manager --disable rht-updates
```

- 3.2. `rht-system` 패키지를 표시하고 설치합니다.

```
[root@servera ~]# dnf list rht-system
Available Packages
rht-system.noarch 1.0.0-1 content.example.com_rhel9.0_x86_64_rhcsa-practice_rht
[root@servera ~]# dnf install rht-system
Dependencies resolved.
=====
Package          Arch      Version       Repository      Size
=====
Installing:
  rht-system      noarch   1.0.0-1      content...._rht  3.7 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
```

```
Installed:
  rht-system-1.0.0-1.noarch
Complete!
```

3.3. **rht-system** 패키지가 설치되고 패키지의 버전 번호를 기록합니다.

```
[root@servera ~]# dnf list rht-system
Installed Packages
rht-system.noarch 1.0.0-1 @content.example.com_rhel9.0_x86_64_rhcsa-practice_rht
```

▶ 4. **rht-updates** 소프트웨어 리포지토리를 활성화하고 관련 소프트웨어 패키지를 모두 업데이트합니다.

4.1. **dnf config-manager --enable**을 사용하여 **rht-updates** 리포지토리를 활성화합니다.

```
[root@servera ~]# dnf config-manager --enable rht-updates
```

4.2. **dnf update**에서 **servera** 명령을 사용하여 모든 소프트웨어 패키지를 업데이트합니다.

```
[root@servera ~]# dnf update
Dependencies resolved.
=====
Package           Arch      Version       Repository      Size
=====
Upgrading:
  rht-system      noarch   1.0.0-2      rht-updates    7.5 k
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

4.3. **rht-system** 패키지가 업그레이드되었는지 확인하고 패키지의 버전 번호를 기록합니다.

```
[root@servera ~]# dnf list rht-system
Installed Packages
rht-system.noarch          1.0.0-2          @rht-updates
```

▶ 5. **servera**를 종료합니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish software-repo
```

이것으로 섹션을 완료합니다.

▶ 랩

소프트웨어 패키지 설치 및 업데이트

이 랩에서는 소프트웨어 리포지토리를 관리하고 해당 리포지토리에서 패키지를 설치 및 업그레이드합니다.

결과

- 소프트웨어 리포지토리를 관리합니다.
- 리포지토리에서 패키지를 설치하고 업그레이드합니다.
- RPM 패키지를 설치합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start software-review
```

지침

1. **serverb** 시스템에서 업데이트를 가져올 소프트웨어 리포지토리를 구성합니다. 리포지토리 이름을 **errata**로 지정하고 **/etc/yum.repos.d/errata.repo** 파일에서 리포지토리를 구성합니다. http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata 리포지토리를 사용하도록 **errata.repo** 파일을 구성합니다. GPG 서명은 확인하지 마십시오.
2. **serverb**에서 **rht-system** 패키지를 설치합니다.
3. 보안상의 이유로 **serverb** 시스템은 용지 프린터에 연결할 수 없어야 합니다. **cups** 패키지를 제거하면 이 결과를 얻을 수 있습니다. 마쳤으면 **root** 쉘을 종료합니다.
4. 시작 스크립트는 **serverb** 시스템의 **/home/student** 디렉터리에 **rhcsa-script-1.0.0-1.noarch.rpm** 패키지를 다운로드합니다. **serverb**에서 **rhcsa-script-1.0.0-1.noarch.rpm** 패키지를 사용할 수 있는지 확인하고 **root** 권한을 사용하여 설치합니다. 패키지가 설치되었는지 확인합니다. **serverb** 시스템을 종료합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade software-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish software-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

소프트웨어 패키지 설치 및 업데이트

이 랩에서는 소프트웨어 리포지토리를 관리하고 해당 리포지토리에서 패키지를 설치 및 업그레이드합니다.

결과

- 소프트웨어 리포지토리를 관리합니다.
- 리포지토리에서 패키지를 설치하고 업그레이드합니다.
- RPM 패키지를 설치합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start software-review
```

지침

1. **serverb** 시스템에서 업데이트를 가져올 소프트웨어 리포지토리를 구성합니다. 리포지토리 이름을 **errata**로 지정하고 **/etc/yum.repos.d/errata.repo** 파일에서 리포지토리를 구성합니다. http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata 리포지토리를 사용하도록 **errata.repo** 파일을 구성합니다. GPG 서명은 확인하지 마십시오.

- 1.1. **student** 사용자로 **serverb** 시스템에 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. 다음 내용으로 **/etc/yum.repos.d/errata.repo** 파일을 생성합니다.

```
[errata]
name=Red Hat Updates
baseurl=http://content.example.com/rhel9.0/x86_64/rhcsa-practice/errata
enabled=1
gpgcheck=0
```

2. **serverb**에서 **rht-system** 패키지를 설치합니다.

- 2.1. **rht-system** 패키지에 사용 가능한 패키지를 표시합니다.

```
[root@serverb ~]# dnf list rht-system
Last metadata expiration check: 0:05:27 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Available Packages
rht-system.noarch      1.0.0-2                  errata
```

2.2. 최신 버전의 **rht-system** 패키지를 설치합니다.

```
[root@serverb ~]# dnf install rht-system
...output omitted...
Total download size: 7.5 k
Installed size: 300
Is this ok [y/N]: y
...output omitted...
Complete!
[root@serverb ~]#
```

3. 보안상의 이유로 **serverb** 시스템은 용지 프린터에 연결할 수 없어야 합니다. **cups** 패키지를 제거하면 이 결과를 얻을 수 있습니다. 마쳤으면 **root** 쉘을 종료합니다.

3.1. 설치된 **cups** 패키지를 표시합니다.

```
[root@serverb ~]# dnf list cups
Last metadata expiration check: 0:08:02 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Installed Packages
cups.x86_64      1:2.3.3op2-13.el9      @rhel-9.0-for-x86_64-appstream-rpms
[root@serverb ~]#
```

3.2. **cups** 패키지를 제거합니다.

```
[root@serverb ~]# dnf remove cups.x86_64
...output omitted...
Remove 46 Packages

Freed space: 94 M
Is this ok [y/N]: y
...output omitted...
Complete!
```

3.3. **root** 쉘을 종료합니다.

```
[root@serverb ~]# exit
[student@serverb ~]$
```

4. 시작 스크립트는 **serverb** 시스템의 **/home/student** 디렉터리에 **rhcsa-script-1.0.0-1.noarch.rpm** 패키지를 다운로드합니다. **serverb**에서 **rhcsa-script-1.0.0-1.noarch.rpm** 패키지를 사용할 수 있는지 확인하고 **root** 권한을 사용하여 설치합니다. 패키지가 설치되었는지 확인합니다. **serverb** 시스템을 종료합니다.

4.1. **serverb**에서 **rhcsa-script-1.0.0-1.noarch.rpm** 패키지를 사용할 수 있는지 확인합니다.

```
[student@serverb ~]$ rpm -q -p rhcsa-script-1.0.0-1.noarch.rpm -i
Name        : rhcsa-script
Version     : 1.0.0
Release     : 1
Architecture: noarch
Install Date: (not installed)
Group       : System
Size        : 593
License     : GPL
Signature   : (none)
Source RPM  : rhcsa-script-1.0.0-1.src.rpm
Build Date  : Wed 23 Mar 2022 08:24:21 AM EDT
Build Host  : localhost
Packager    : Bernardo Gargallo
URL         : http://example.com
Summary     : RHCSA Practice Script
Description :
A RHCSA practice script.
The package changes the motd.
```

4.2. rhcsa-script-1.0.0-1.noarch.rpm 패키지를 설치합니다.

```
[student@serverb ~]$ sudo dnf install \
rhcsa-script-1.0.0-1.noarch.rpm
[sudo] password for student: student
Last metadata expiration check: 0:11:06 ago on Wed 27 Apr 2022 05:01:59 AM EDT.
Dependencies resolved.
=====
Package           Architecture      Version       Repository      Size
=====
Installing:
  rhcsa-script    noarch          1.0.0-1      @commandline   7.5 k

Transaction Summary
=====
Install 1 Package

Total size: 7.5 k
Installed size: 593
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing           :                               1/1
  Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
  Installing        : rhcsa-script-1.0.0-1.noarch 1/1
  Running scriptlet: rhcsa-script-1.0.0-1.noarch 1/1
  Verifying          : rhcsa-script-1.0.0-1.noarch 1/1

Installed:
```

```
rhcsa-script-1.0.0-1.noarch
```

```
Complete!
```

4.3. 패키지가 설치되었는지 확인합니다.

```
[student@serverb ~]$ rpm -q rhcsa-script
rhcsa-script-1.0.0-1.noarch
[student@serverb ~]$
```

4.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade software-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish software-review
```

이것으로 섹션을 완료합니다.

요약

- Red Hat Subscription Management는 제품 서브스크립션 자격을 시스템에 부여하고, 소프트웨어 패키지 업데이트를 가져오고, 시스템이 사용하는 지원 계약 및 서브스크립션에 대한 정보를 추적하기 위한 툴을 제공합니다.
- **dnf** 유ти리티는 소프트웨어 패키지를 설치, 업데이트, 제거, 쿼리하기 위한 강력한 명령줄 터입니다.
- Red Hat Enterprise Linux에서는 애플리케이션 스트림을 사용하여 여러 버전의 애플리케이션 패키지 및 해당 종속성을 호스팅할 단일 리포지토리를 제공합니다.

9장

기본 스토리지 관리

목적

명령줄에서 스토리지 장치, 파티션, 파일 시스템 및 스왑 파일을 생성하고 관리합니다.

목표

- 파일 시스템 계층 구조에서 파일 시스템을 추가 및 제거하여 파일 시스템 내용에 액세스합니다.
- 스토리지 파티션을 생성하고, 파일 시스템으로 포맷한 다음, 사용하기 위해 마운트합니다.
- 스왑 공간을 생성하고 관리하여 실제 메모리를 확보합니다.

섹션

- 파일 시스템 마운트 및 마운트 해제(안내에 따른 연습)
- 파티션, 파일 시스템, 영구 마운트 추가(안내에 따른 연습)
- 스왑 공간 관리(안내에 따른 연습)

랩

- 기본 스토리지 관리

파일 시스템 마운트 및 마운트 해제

목표

파일 시스템 계층 구조에서 파일 시스템을 추가 및 제거하여 파일 시스템 내용에 액세스합니다.

수동으로 파일 시스템 마운트

이동식 스토리지 장치의 파일 시스템에 액세스하려면 스토리지 장치를 마운트해야 합니다. **mount** 명령을 사용하면 **root** 사용자가 파일 시스템을 수동으로 마운트할 수 있습니다. **mount** 명령의 첫 번째 인수는 마운트할 파일 시스템을 지정합니다. 두 번째 인수는 파일 시스템 계층 구조의 마운트 지점으로 디렉터리를 지정합니다.

mount 명령을 사용하여 다음 방법 중 하나로 파일 시스템을 마운트할 수 있습니다.

- **/dev** 디렉터리의 장치 파일 이름 사용
- 장치의 UUID(Universally Unique Identifier) 사용

마운트할 장치를 식별하고 마운트 지점이 있는지 확인한 다음, 마운트 지점에 장치를 마운트합니다.

블록 장치 식별

서버의 HDD(하드 디스크 드라이브) 또는 SSD(Solid-State Device)이건, USB 스토리지 장치건 관계없이 핫플러그형 스토리지 장치는 매번 시스템의 다른 포트에 연결될 수 있습니다. **lsblk** 명령을 사용하여 지정된 블록 장치 또는 사용 가능한 모든 장치의 세부 정보를 표시합니다.

```
[root@host ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
vda    252:0    0  10G  0 disk
└─vda1 252:1    0   1M  0 part
└─vda2 252:2    0 200M  0 part /boot/efi
└─vda3 252:3    0 500M  0 part /boot
└─vda4 252:4    0 9.3G  0 part /
vdb    252:16   0   5G  0 disk
vdc    252:32   0   5G  0 disk
vdd    252:48   0   5G  0 disk
```

파티션 크기는 파티션 이름을 알 수 없을 때 장치를 식별하는 데 도움이 됩니다. 예를 들어 이전 출력에서 식별된 파티션의 크기가 9.3GB이면 **/dev/vda4** 파티션을 마운트합니다.

파티션 이름을 사용하여 파일 시스템 마운트

다음 예제에서는 **/mnt/data** 마운트 지점에 **/dev/vda4** 파티션을 마운트합니다.

```
[root@host ~]# mount /dev/vda4 /mnt/data
```

파일 시스템을 마운트하기 전에 마운트 지점 디렉터리가 있어야 합니다. 임시 마운트 지점으로 사용할 수 있는 **/mnt** 디렉터리가 있습니다.

**중요**

마운트 지점으로 사용할 디렉터리가 비어 있지 않으면 파일 시스템이 마운트되는 동안 기존 파일은 숨겨지고 액세스할 수 없습니다. 마운트된 파일 시스템을 마운트 해제하면 원본 파일에 다시 액세스할 수 있습니다.

장치 탐지 순서 및 스토리지 장치 이름 지정은 시스템에서 장치를 추가하거나 제거할 때 변경될 수 있습니다. 마운트 파일 시스템에 대해 변경되지 않는 장치 식별자를 일관성 있게 사용하는 것이 좋습니다.

파티션 UUID를 사용하여 파일 시스템 마운트

파일 시스템과 연결된 안정적인 식별자 중 하나가 UUID(Universally Unique Identifier)입니다. UUID는 파일 시스템 수퍼 블록에 저장되며 파일 시스템을 재생성할 때까지 동일하게 유지됩니다.

`lsblk -fp` 명령은 장치의 전체 경로, UUID 및 마운트 지점, 파티션의 파일 시스템 유형을 표시합니다. 파일 시스템이 마운트되지 않은 경우 마운트 지점은 비어 있습니다.

```
[root@host ~]# lsblk -fp
  NAME      FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE% MOUNTPOINTS
  /dev/vda
    └─/dev/vda1
      ├─/dev/vda2 vfat    FAT16     7B77-95E7          192.3M    4% /boot/efi
      ├─/dev/vda3 xfs      boot    2d67e6d0-...-1f091bf1  334.9M   32% /boot
      └─/dev/vda4 xfs      root    efd314d0-...-ae98f652    7.7G   18% /
  /dev/vdb
  /dev/vdc
  /dev/vdd
```

파일 시스템 UUID로 파일 시스템을 마운트합니다.

```
[root@host ~]# mount UUID="efd314d0-b56e-45db-bbb3-3f32ae98f652" /mnt/data
```

자동으로 이동식 스토리지 장치 마운트

그래픽 데스크톱 환경을 사용하는 경우 미디어가 있는 것이 탐지되면 시스템에서 이동식 스토리지 미디어를 자동으로 마운트합니다.

이동식 스토리지 장치는 `/run/media/USERNAME/LABEL` 위치에 마운트됩니다. USERNAME은 그래픽 환경에 로그인한 사용자의 이름입니다. LABEL은 식별자로, 일반적으로 스토리지 미디어의 레이블입니다.

이동식 장치를 안전하게 분리하려면 먼저 장치의 모든 파일 시스템을 수동으로 마운트 해제합니다.

파일 시스템 마운트 해제

시스템 종료 및 재부팅 절차를 수행하면 모든 파일 시스템이 자동으로 마운트 해제됩니다. 파일 시스템 데이터 무결성을 보장하기 위해 모든 파일 시스템 데이터가 스토리지 장치로 플러시됩니다.

**경고**

파일 시스템 데이터는 정상 작동 중에 메모리 캐시를 사용합니다. 드라이브를 분리하기 전에 이동식 드라이브의 파일 시스템을 마운트 해제해야 합니다. 마운트 해제 절차를 수행하면 드라이브를 릴리스하기 전에 데이터가 디스크로 플러시됩니다.

umount 명령은 마운트 지점을 인수로 사용하여 파일 시스템을 마운트 해제합니다.

```
[root@host ~]# umount /mnt/data
```

마운트된 파일 시스템이 사용 중이면 마운트 해제할 수 없습니다. **umount** 명령이 성공하려면 모든 프로세스에서 마운트 지점 아래의 데이터에 대한 액세스를 중지해야 합니다.

다음 예제에서는 쉘이 **/mnt/data** 디렉터리를 현재 작업 디렉터리로 사용하므로 **umount** 명령이 실패하고 오류 메시지를 생성합니다.

```
[root@host ~]# cd /mnt/data
[root@host data]# umount /mnt/data
umount: /mnt/data: target is busy.
```

lsof 명령은 열려 있는 모든 파일과 파일 시스템에 액세스하는 프로세스를 표시합니다. 이 목록은 파일 시스템을 마운트 해제할 수 없게 하는 프로세스를 식별하는데 도움이 됩니다.

```
[root@host data]# lsof /mnt/data
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
bash    1593 root cwd DIR 253,17      6  128 /mnt/data
lsof    2532 root cwd DIR 253,17     19  128 /mnt/data
lsof    2533 root cwd DIR 253,17     19  128 /mnt/data
```

프로세스를 식별하고 프로세스가 완료될 때까지 기다리거나 **SIGTERM** 또는 **SIGKILL** 신호를 전송하여 프로세스를 종료합니다. 이 경우에는 마운트 지점 밖에 있는 현재 작업 디렉터리로 변경하는 것으로 충분합니다.

```
[root@host data]# cd
[root@host ~]# umount /mnt/data
```



참조

lsblk(8), **mount(8)**, **umount(8)**, **lsof(8)** 도움말 페이지

▶ 연습 가이드

파일 시스템 마운트 및 마운트 해제

이 연습에서는 파일 시스템을 마운트 및 마운트 해제합니다.

결과

- 새 파일 시스템을 확인하고 지정된 마운트 지점에 마운트한 다음, 파일 시스템을 마운트 해제합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start fs-mount
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 파일 시스템이 있는 새 파티션이 **servera** 시스템의 **/dev/vdb** 디스크에 추가되었습니다. UUID를 사용하여 사용 가능한 새 파티션을 **/mnt/part1** 마운트 지점에 마운트합니다.

- 2.1. **/mnt/part1** 디렉터리를 생성합니다.

```
[root@servera ~]# mkdir /mnt/part1
```

- 2.2. **/dev/vdb1** 장치의 UUID를 쿼리합니다.

```
[root@servera ~]# lsblk -fp /dev/vdb
NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
/dev/vdb
└─/dev/vdb1 xfs   a04c511a-b805-4ec2-981f-42d190fc9a65
```

- 2.3. UUID를 사용하여 파일 시스템을 **/mnt/part1** 디렉터리에 마운트합니다. 이전 명령 출력의 **/dev/vdb1** UUID를 사용합니다.

```
[root@servera ~]# mount \
UUID="a04c511a-b805-4ec2-981f-42d190fc9a65" /mnt/part1
```

2.4. `/dev/vdb1` 장치가 `/mnt/part1` 디렉터리에 마운트되었는지 확인합니다.

```
[root@servera ~]# lsblk -fp /dev/vdb
  NAME      FSTYPE LABEL UUID                                     MOUNTPOINT
  /dev/vdb
  └─/dev/vdb1 xfs   a04c511a-b805-4ec2-981f-42d190fc9a65 /mnt/part1
```

- ▶ 3. `/mnt/part1` 디렉터리로 변경하고 `testdir` 하위 디렉터리를 생성합니다. `/mnt/part1/testdir/newmount` 파일을 생성합니다.

3.1. `/mnt/part1` 디렉터리로 변경합니다.

```
[root@servera ~]# cd /mnt/part1
```

3.2. `/mnt/part1/testdir` 디렉터리를 생성합니다.

```
[root@servera part1]# mkdir testdir
```

3.3. `/mnt/part1/testdir/newmount` 파일을 생성합니다.

```
[root@servera part1]# touch testdir/newmount
```

- ▶ 4. `/mnt/part1` 디렉터리에 마운트된 파일 시스템을 마운트 해제합니다.

4.1. 쉘이 `/mnt/part1` 디렉터리에 있는 동안 `/mnt/part1` 디렉터리를 마운트 해제합니다. `umount` 명령으로는 장치를 마운트 해제하지 못합니다.

```
[root@servera part1]# umount /mnt/part1
umount: /mnt/part1: target is busy.
```

4.2. 쉘의 현재 디렉터리를 `/root` 디렉터리로 변경합니다.

```
[root@servera part1]# cd
[root@servera ~]#
```

4.3. `/mnt/part1` 디렉터리를 마운트 해제합니다.

```
[root@servera ~]# umount /mnt/part1
```

- ▶ 5. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish fs-mount
```

이것으로 섹션을 완료합니다.

파티션, 파일 시스템, 영구 마운트 추가

목표

스토리지 파티션을 생성하고, 파일 시스템으로 포맷한 다음, 사용하기 위해 마운트합니다.

디스크 파티셔닝

디스크 파티셔닝은 하드 드라이브를 여러 개의 논리 스토리지 파티션으로 나눕니다. 파티션을 사용하면 다양한 요구 사항에 따라 스토리지를 분할할 수 있으며, 이러한 분할은 다양한 이점을 제공합니다.

- 애플리케이션 또는 사용자가 사용할 수 있는 공간을 제한합니다.
- 사용자 파일에서 운영 체제와 프로그램 파일을 구분합니다.
- 메모리 스왑을 위해 별도 영역을 생성합니다.
- 디스크 공간 사용을 제한하여 진단 도구 및 백업 이미징의 성능을 향상합니다.

MBR 파티션 체계

MBR(Master Boot Record) 파티셔닝 체계는 BIOS 펌웨어를 실행하는 시스템의 표준입니다. 이 스키마는 최대 네 개의 주 파티션을 지원합니다. Linux 시스템에서는 확장 및 논리 파티션을 사용하여 최대 15개의 파티션을 생성할 수 있습니다. 파티션 크기가 32비트인 경우 MBR로 파티셔닝되는 디스크 크기는 최대 2TiB일 수 있습니다.

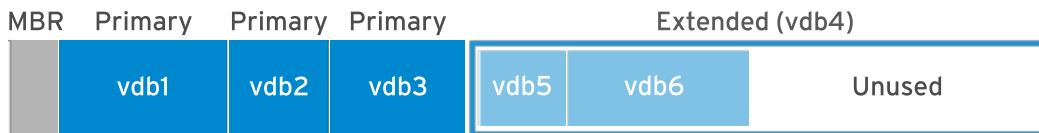


그림 9.1: /dev/vdb 스토리지 장치의 MBR 파티셔닝

2TiB의 디스크 및 파티션 크기 제한은 이제 일반적이고 제한적인 제한입니다. 결과적으로 레거시 MBR 체계가 GPT(GUID Partition Table) 파티셔닝 체계로 대체되었습니다.

GPT 파티션 테이블

UEFI(Unified Extensible Firmware Interface) 펌웨어를 실행하는 시스템의 경우 GPT는 디스크 파티셔닝의 표준이며 MBR 체계의 제한 사항을 처리합니다. GPT는 최대 128개의 파티션을 제공합니다. GPT 체계는 논리 블록 주소에 64비트를 할당하여 최대 8제비바이트(ZiB) 또는 80억 테비바이트(TiB)의 파티션 및 디스크를 지원합니다.



그림 9.2: /dev/vdb 스토리지 장치의 GPT 파티셔닝

GPT 파티셔닝은 MBR에 비해 더 많은 기능과 이점을 제공합니다. GPT는 GUID(전역 고유 식별자)를 사용하여 각 디스크와 파티션을 식별합니다. GPT는 기본 GPT를 디스크 헤드에, 백업 보조 GPT를 디스크 끝부분에

배치하여 파티션 테이블을 이중화합니다. GPT는 체크섬을 사용하여 GPT 헤더와 파티션 테이블에서 오류를 감지합니다.

파티션 관리

관리자는 파티션 편집기 프로그램을 사용하여 파티션 생성, 파티션 삭제, 파티션 유형 변경 등 디스크의 파티션을 변경할 수 있습니다.

Red Hat Enterprise Linux 명령줄의 표준 파티션 편집기는 **parted**입니다. MBR 파티셔닝 체계 또는 GPT 파티셔닝 체계를 사용하는 스토리지와 함께 **parted** 파티션 편집기를 사용할 수 있습니다.

parted 명령은 수정할 전체 스토리지 장치 또는 디스크를 나타내는 장치 이름을 첫 번째 인수로 사용하고 그 뒤에 하위 명령을 사용합니다. 다음 예에서는 **print** 하위 명령을 사용하여 **/dev/vda** 블록 장치(시스템에서 감지한 첫 번째 '가상화된 I/O 디스크')인 디스크의 파티션 테이블을 표시합니다.

```
[root@host ~]# parted /dev/vda print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  10.7GB  10.7GB  primary   xfs          boot
 2      10.7GB   53.7GB  42.9GB  primary   xfs
```

대화형 파티셔닝 세션을 열려면 하위 명령 없이 **parted** 명령을 사용합니다.

```
[root@host ~]# parted /dev/vda
GNU Parted 3.4
Using /dev/vda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 53.7GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1      1049kB  10.7GB  10.7GB  primary   xfs          boot
 2      10.7GB   53.7GB  42.9GB  primary   xfs

(parted) quit
[root@host ~]#
```

기본적으로 **parted** 명령은 10의 거듭제곱으로 크기를 표시합니다(KB, MB, GB). 다음 값을 허용하는 **unit** 매개 변수를 사용하여 유닛 크기를 변경할 수 있습니다.

- 섹터의 경우 **s**
- 바이트의 경우 **B**
- **MiB**, **GiB** 또는 **TiB** (2의 거듭제곱)
- **MB**, **GB** 또는 **TB** (10의 거듭제곱)

```
[root@host ~]# parted /dev/vda unit s print
Model: Virtio Block Device (virtblk)
Disk /dev/vda: 104857600s
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Partition Flags:

Number  Start      End        Size       Type      File system  Flags
 1      2048s    20971486s  20969439s  primary   xfs          boot
 2      20971520s 104857535s  83886016s  primary   xfs
```

위 예제와 같이 같은 줄에 여러 하위 명령을 지정할 수도 있습니다(여기서는 **unit** 및 **print**).

새 디스크에 파티션 테이블 작성

새 드라이브를 파티셔닝하려면 먼저 디스크 레이블을 작성하십시오. 디스크 레이블은 사용할 파티셔닝 체계를 나타냅니다. **parted**를 사용하여 MBR 디스크 레이블 또는 GPT 디스크 레이블을 작성합니다.

```
[root@host ~]# parted /dev/vdb mklabel msdos
[root@host ~]# parted /dev/vdb mklabel gpt
```



경고

mklabel 하위 명령은 기존 파티션 테이블을 지웁니다. 기존 데이터에 관계없이 디스크를 재사용하려는 경우 **mklabel** 하위 명령을 사용합니다. 새 레이블이 파티션 경계를 변경하는 경우 기존 파일 시스템의 모든 데이터에 액세스할 수 없게 됩니다.

MBR 파티션 생성

다음 지침은 MBR 디스크 파티션을 생성합니다. 파티션을 만들 디스크 장치를 지정합니다.

parted 명령을 실행하고 디스크 장치 이름을 인수로 지정하여 대화형 모드로 시작합니다. 세션에서 (**parted**)를 하위 명령 프롬프트로 표시합니다.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

mkpart 하위 명령을 사용하여 주 파티션이나 확장 파티션을 생성합니다.

```
(parted) mkpart
Partition type? primary/extended? primary
```



참고

MBR 파티셔닝 디스크에 파티션이 5개 이상 필요한 경우 주 파티션 3개와 확장 파티션 1개를 생성합니다. 이 확장 파티션은 여러 논리 파티션을 생성할 수 있는 컨테이너 역할을 합니다.

9장 | 기본 스토리지 관리

파티션에 생성하려는 파일 시스템 유형을 나타냅니다(예: **xfs** 또는 **ext4**). 이 값은 파일 시스템을 생성하지 않으면 단지 유용한 파티션 유형 레이블입니다.

```
File system type? [ext2]? xfs
```

지원되는 파일 시스템 유형을 나열하려면 다음 명령을 사용합니다.

```
[root@host ~]# parted /dev/vdb help mkpart
...output omitted...
mkpart PART-TYPE [FS-TYPE] START END      make a partition

PART-TYPE is one of: primary, logical, extended
FS-TYPE is one of: udf, btrfs, nilfs2, ext4, ext3, ext2, f2fs, fat32, fat16,
hfsx, hfs+, hfs, jfs, swsusp, linux-swap(v1), linux-swap(v0), ntfs,
reiserfs, hp-ufs, sun-ufs, xfs, apfs2, apfs1, asfs, amufs5, amufs4, amufs3,
amufs2, amufs1, amufs0, amufs, affs7, affs6, affs5, affs4, affs3, affs2,
affs1, affs0, linux-swap, linux-swap(new), linux-swap(old)

'mkpart' makes a partition without creating a new file system on the
partition. FS-TYPE may be specified to set an appropriate partition
ID.
```

새 파티션을 시작할 디스크 섹터를 지정합니다.

```
Start? 2048s
```

s 접미사는 섹터 단위로 값을 제공하거나 **MiB**, **GiB**, **TiB**, **MB**, **GB** 또는 **TB** 접미사를 사용합니다. **parted** 명령에는 기본적으로 **MB** 접미사가 사용됩니다. **parted** 명령은 제공된 값을 반올림하여 디스크 제한 조건을 충족합니다.

parted 명령이 시작되면 장치에서 디스크 토플로지를 검색합니다(예: 디스크의 실제 블록 크기). **parted** 명령은 제공하는 시작 위치가 성능을 최적화하도록 파티션을 디스크 구조에 올바르게 정렬하는지 확인합니다. 시작 위치로 인해 파티션이 잘못 정렬되는 경우 **parted** 명령에서 경고를 표시합니다. 대부분의 디스크에서 2048의 배수인 시작 섹터는 안전합니다.

새 파티션을 끝낼 디스크 섹터를 지정하고 **parted**를 종료합니다. 종료를 크기 또는 끝 위치로 지정할 수 있습니다.

```
End? 1000MB
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

끝 위치를 제공하면 **parted** 명령이 새 파티션 정보로 디스크의 파티션 테이블을 업데이트합니다.

udevadm settle 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 감지하고 **/dev** 디렉터리에 관련 장치 파일을 생성할 때까지 기다립니다. 작업이 완료되면 프롬프트가 반환됩니다.

```
[root@host ~]# udevadm settle
```

대화형 모드의 대안으로 단일 명령에서 파티션을 생성할 수 있습니다.

```
[root@host ~]# parted /dev/vdb mkpart primary xfs 2048s 1000MB
```

GPT 파티션 생성

GPT 체계에서도 **parted** 명령을 사용하여 새 파티션을 생성합니다. 파티션을 만들 디스크 장치를 지정합니다.

root 사용자로 **parted** 명령을 실행하고 인수로 디스크 장치 이름을 지정합니다.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

mkpart 하위 명령을 사용하여 파티션 생성을 시작합니다. GPT 체계에서는 각 파티션에 이름이 지정됩니다.

```
(parted) mkpart
Partition name? []? userdata
```

파티션에 생성하려는 파일 시스템 유형을 나타냅니다(예: **xfs** 또는 **ext4**). 이 값은 파일 시스템을 생성하지 않지만 유용한 파티션 유형 레이블입니다.

```
File system type? [ext2]? xfs
```

새 파티션을 시작할 디스크 섹터를 지정합니다.

```
Start? 2048s
```

새 파티션을 끝낼 디스크 섹터를 지정하고 **parted**를 종료합니다. 끝 위치를 제공하면 **parted** 명령이 새 파티션 세부 정보로 디스크의 GPT를 업데이트합니다.

```
End? 1000MB
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

udevadm settle 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 감지하고 **/dev** 디렉터리에 관련 장치 파일을 생성할 때까지 기다립니다. 작업이 완료되면 프롬프트가 반환됩니다.

```
[root@host ~]# udevadm settle
```

대화형 모드의 대안으로 단일 명령에서 파티션을 생성할 수 있습니다.

```
[root@host ~]# parted /dev/vdb mkpart userdata xfs 2048s 1000MB
```

파티션 삭제

다음 지침은 MBR 및 GPT 파티셔닝 체계에 모두 적용됩니다. 제거할 파티션이 포함된 디스크를 지정합니다. 디스크 장치를 유일한 인수로 사용하여 **parted** 명령을 실행합니다.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted)
```

삭제할 파티션의 파티션 번호를 식별합니다.

```
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name      Flags
 1      1049kB  1000MB  999MB  xfs          usersdata
```

파티션을 삭제하고 **parted**를 종료합니다. **rm** 하위 명령은 디스크의 파티션 테이블에서 즉시 파티션을 삭제합니다.

```
(parted) rm 1
(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

대화형 모드의 대안으로 단일 명령에서 파티션을 삭제할 수 있습니다.

```
[root@host ~]# parted /dev/vdb rm 1
```

파일 시스템 생성

블록 장치를 생성한 후 다음 단계는 블록 장치에 파일 시스템을 추가하는 것입니다. Red Hat Enterprise Linux는 다양한 파일 시스템 유형을 지원하며 권장되는 기본값은 XFS입니다.

root 사용자로 **mkfs.xfs** 명령을 사용하여 블록 장치에 XFS 파일 시스템을 적용합니다. ext4 파일 시스템의 경우 **mkfs.ext4** 명령을 사용합니다.

```
[root@host ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=60992 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=1       finobt=1, sparse=1, rmapbt=0
                                =                      reflink=1  bigtime=1 inobtcount=1
data     =                      bsize=4096   blocks=243968, imaxpct=25
                                =                      sunit=0     swidth=0 blks
```

```

naming      =version 2          bsize=4096   ascii-ci=0, ftype=1
log         =internal log       bsize=4096   blocks=1566, version=2
        =                           sectsz=512  sunit=0 blks, lazy-count=1
realtime    =none               extsz=4096  blocks=0, rtextents=0

```

파일 시스템 마운트

파일 시스템을 추가한 후 마지막 단계는 디렉터리 구조의 디렉터리에 파일 시스템을 마운트하는 것입니다. 디렉터리 계층에 파일 시스템을 마운트하면 사용자 공간 유ти리티가 장치의 파일에 액세스하거나 쓸 수 있습니다.

파일 시스템 수동 마운트

마운트 지점 디렉터리 위치에 장치를 수동으로 연결하려면 **mount** 명령을 사용합니다. **mount** 명령에는 장치 및 마운트 지점이 필요하며 파일 시스템 마운트 옵션이 포함될 수 있습니다. 파일 시스템 옵션은 파일 시스템 동작을 사용자 지정합니다.

```
[root@host ~]# mount /dev/vdb1 /mnt
```

또한 **mount** 명령을 사용하여 현재 마운트된 파일 시스템, 마운트 지점 및 해당 옵션을 확인합니다.

```
[root@host ~]# mount | grep vdb1
/dev/vdb1 on /mnt type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

파일 시스템 영구 마운트

파일 시스템을 수동으로 마운트할 경우 포맷된 장치가 액세스 가능하며 예상한 대로 작동하는지 쉽게 확인할 수 있습니다. 그러나 서버가 재부팅되면 시스템에서 파일 시스템을 다시 자동으로 마운트하지 않습니다.

시스템 부팅 시 파일 시스템이 자동으로 마운트되도록 구성하려면 **/etc/fstab** 파일에 항목을 추가합니다. 이 구성 파일은 시스템 부팅 시 마운트할 파일 시스템을 나열합니다.

/etc/fstab 파일은 공백으로 구분된 파일이며 행당 여섯 개의 필드가 있습니다.

```
[root@host ~]# cat /etc/fstab

#
# /etc/fstab
# Created by anaconda on Thu Apr 5 12:05:19 2022
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
UUID=a8063676-44dd-409a-b584-68be2c9f5570   /          xfs  defaults  0 0
UUID=7a20315d-ed8b-4e75-a5b6-24ff9e1f9838   /dbdata    xfs  defaults  0 0
```

첫 번째 필드는 장치를 지정합니다. 이 예제에서는 UUID를 사용하여 장치를 지정합니다. 파일 시스템이 UUID를 생성하고 생성 시 수퍼 블록에 저장합니다. 또는 **/dev/vdb1** 등의 장치 파일을 사용할 수 있습니다.

두 번째 필드는 디렉터리 구조에서 블록 장치에 액세스할 수 있는 디렉터리 마운트 지점입니다. 마운트 지점이 있어야합니다. 그렇지 않은 경우 **mkdir** 명령으로 마운트 지점을 생성합니다.

세 번째 필드는 **xfs** 또는 **ext4**와 같은 파일 시스템 유형입니다.

네 번째 필드는 장치에 적용할 쉼표로 구분된 옵션 목록입니다. **defaults**는 일반적으로 사용되는 옵션 집합입니다. **mount(8)** 도움말 페이지는 사용 가능한 다른 옵션을 문서화합니다.

다섯 번째 필드는 **dump** 명령이 장치를 백업하는 데 사용됩니다. 다른 백업 애플리케이션은 일반적으로 이 필드를 사용하지 않습니다.

마지막 필드인 **fsck** 순서 필드는 시스템 부팅 시 **fsck** 명령을 실행하여 파일 시스템이 정리되었음을 확인해야 하는지를 결정합니다. 이 필드의 값은 **fsck**의 실행 순서를 나타냅니다. XFS 파일 시스템의 경우 XFS에서 파일 시스템 상태를 확인하는 데 **fsck**를 사용하여 않으므로 이 필드를 **0**으로 설정합니다. ext4 파일 시스템의 경우 루트 파일 시스템은 **1**로 설정하고 기타 ext4 파일 시스템은 **2**로 설정합니다. 이 표기법을 사용하면 **fsck** 유ти리티에서 루트 파일 시스템을 먼저 처리한 후 개별 디스크의 파일 시스템은 동시에 점검하고 동일한 디스크의 파일 시스템은 순서대로 점검합니다.



참고

/etc/fstab에 잘못된 항목이 있으면 시스템이 부팅되지 않을 수 있습니다. 새 파일 시스템을 수동으로 마운트 해제한 다음 **mount /mountpoint**를 사용하여 **/etc/fstab** 파일을 읽고 해당 항목의 마운트 옵션으로 파일 시스템을 다시 마운트하여 항목이 유효한지 확인합니다. **mount** 명령이 오류를 반환할 경우 시스템을 재부팅하기 전에 수정합니다.

또는 파티션 유용성을 위해 **findmnt --verify** 명령을 사용하여 **/etc/fstab** 파일을 구문 분석합니다.

/etc/fstab 파일에 항목을 추가하거나 제거할 때는 **systemd** 데몬이 새 구성으로드하여 사용하도록 **systemctl daemon-reload** 명령을 실행하거나 서버를 재부팅합니다.

```
[root@host ~]# systemctl daemon-reload
```

클라우드 프로바이더가 가상 시스템의 기본 스토리지 계층을 변경하거나 시스템 부팅 시 디스크가 다른 순서로 감지되는 경우와 같이 블록 장치 이름이 특정 시나리오에서 변경될 수 있으므로 파일 시스템을 영구적으로 마운트하는 데 UUID를 사용하는 것이 좋습니다. 블록 장치 파일 이름은 변경될 수 있지만 UUID는 파일 시스템의 수퍼 블록에 일관되게 유지됩니다.

lsblk --fs 명령을 사용하여 시스템에 연결된 블록 장치를 스캔하고 파일 시스템 UUID를 검색합니다.

```
[root@host ~]# lsblk --fs
NAME   FSTYPE  FSVER  LABEL      UUID          FSAVAIL FSUSE% MOUNTPOINTS
vda
└─vda1
└─vda2  xfs    boot    49dd...75fdf  312M    37%    /boot
└─vda3  xfs    root    8a90...ce0da  4.8G    48%    /
```



참조

`info parted`(GNU Parted User Manual)

`parted(8)`, `mkfs(8)`, `mount(8)`, `lsblk(8)` 및 `fstab(5)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/managing_file_systems/index

에 있는 Configuring and Managing File Systems 가이드를 참조하십시오.

▶ 연습 가이드

파티션, 파일 시스템, 영구 마운트 추가

이 연습에서는 새 스토리지 장치에 파티션을 생성하고 XFS 파일 시스템으로 포맷한 다음 부팅 시 마운트되도록 구성하고 사용할 수 있도록 마운트합니다.

결과

- parted, mkfs.xfs** 및 기타 명령을 사용하여 새 디스크에 파티션을 생성하고 포맷한 다음 영구적으로 마운트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start storage-partitions
```

지침

- ▶ 1. **servera**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **/dev/vdb** 장치에 **msdos** 디스크 레이블을 생성합니다.

```
[root@servera ~]# parted /dev/vdb mklabel msdos
Information: You may need to update /etc/fstab.
```

- ▶ 3. 1GB의 기본 파티션을 추가합니다. 정확한 정렬을 위해 2048 섹터에서 파티션을 시작합니다. 파티션 파일 시스템 유형을 XFS로 설정합니다.

- 3.1. **parted** 대화형 모드를 사용하여 파티션을 생성합니다.

```
[root@servera ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mkpart
Partition type? primary/extended? primary
File system type? [ext2]? xfs
Start? 2048s
```

```
End? 1001MB
(parted) quit
Information: You may need to update /etc/fstab.
```

파티션이 2048 섹터에서 시작하기 때문에 위 명령은 파티션 크기 1000MB(1GB)를 확보하기 위해 끝 위치를 1001MB로 설정합니다.

또는 비 대화형 명령 `parted /dev/vdb mkpart primary xfs 2048s 1001 MB`를 사용하여 동일한 작업을 수행할 수 있습니다.

3.2. /dev/vdb 장치의 파티션을 나열하여 작업을 확인합니다.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type      File system  Flags
 1       1049kB  1001MB  1000MB  primary
```

3.3. udevadm settle 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 등록할 때까지 기다린 후 완료되면 반환됩니다.

```
[root@servera ~]# udevadm settle
```

▶ 4. 새 파티션을 XFS 파일 시스템으로 포맷합니다.

```
[root@servera ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=61056 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=1      finobt=1, sparse=1, rmapbt=0
                                =                      reflink=1 bigtime=1 inobtcount=1
data     =                      bsize=4096   blocks=244224, imaxpct=25
                                =                      sunit=0    swidth=0 blks
naming  =version 2              bsize=4096   ascii-ci=0, ftype=1
log     =internal log          bsize=4096   blocks=1566, version=2
                                =                      sectsz=512  sunit=0 blks, lazy-count=1
realtime =none                  extsz=4096   blocks=0, rtextents=0
```

▶ 5. /archive 디렉터리에 영구적으로 마운트되도록 새 파일 시스템을 구성합니다.

5.1. /archive 디렉터리를 생성합니다.

```
[root@servera ~]# mkdir /archive
```

5.2. /dev/vdb1 장치의 UUID를 검색합니다. 출력의 UUID는 시스템마다 다를 수 있습니다.

```
[root@servera ~]# lsblk --fs /dev/vdb
  NAME   FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
  MOUNTPOINTS
vdb
└─vdb1 xfs            881e856c-37b1-41e3-b009-ad526e46d987
```

- 5.3. **/etc/fstab** 파일에 다음 항목을 추가합니다. UUID를 위 단계에서 검색한 UUID로 바꿉니다.

```
...output omitted...
UUID=881e856c-37b1-41e3-b009-ad526e46d987 /archive xfs defaults 0 0
```

- 5.4. 시스템의 **systemd** 데몬을 업데이트하여 새 **/etc/fstab** 파일 구성을 등록합니다.

```
[root@servera ~]# systemctl daemon-reload
```

- 5.5. 새 항목이 있는 새 파일 시스템을 **/etc/fstab** 파일에 마운트합니다.

```
[root@servera ~]# mount /archive
```

- 5.6. **/archive** 디렉터리에 새 파일 시스템이 마운트되었는지 확인합니다.

```
[root@servera ~]# mount | grep /archive
/dev/vdb1 on /archive type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- ▶ 6. **servera**를 재부팅합니다. 서버가 재부팅되면 로그인한 다음 **/dev/vdb1** 장치가 **/archive** 디렉터리에 마운트되었는지 확인합니다. 완료되면 **servera**에서 로그아웃합니다.

- 6.1. **servera**를 재부팅합니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 6.2. **servera**가 재부팅될 때까지 기다린 후 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 6.3. **/dev/vdb1** 장치가 **/archive** 디렉터리에 마운트되었는지 확인합니다.

```
[student@servera ~]$ mount | grep /archive
/dev/vdb1 on /archive type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

- 6.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish storage-partitions
```

이것으로 섹션을 완료합니다.

스왑 공간 관리

목표

스왑 공간을 생성하고 관리하여 실제 메모리를 확보합니다.

스왑 공간의 개념

스왑 공간은 Linux 커널 메모리 관리 하위 시스템에서 제어하는 디스크 영역입니다. 커널은 메모리에 비활성 페이지를 보관하여 시스템 RAM을 보완하기 위해 스왑 공간을 사용합니다. 시스템의 가상 메모리에는 결합된 시스템 RAM과 스왑 공간이 포함됩니다.

시스템의 메모리 사용량이 정의된 한도를 초과할 경우 커널은 RAM에서 프로세스에 할당된 유휴 메모리 페이지를 검색합니다. 커널은 유휴 페이지를 스왑 영역에 쓰고 RAM 페이지를 다른 프로세스에 다시 할당합니다. 프로그램에서 디스크의 페이지에 액세스해야 하는 경우 커널은 메모리의 다른 유휴 페이지를 찾아 디스크에 쓴 다음 스왑 영역에서 필요한 페이지를 불러옵니다.

스왑 영역이 디스크에 있으므로 스왑이 RAM에 비해 속도가 느립니다. 스왑 공간은 시스템 RAM을 늘리는 데 사용되지만, 워크로드에 비해 RAM이 부족한 경우 스왑 공간을 지속 가능한 해결책으로 간주해서는 안 됩니다.

스왑 공간 계산

관리자는 시스템의 메모리 워크로드에 따라 스왑 공간의 크기를 조정해야 합니다. 애플리케이션 벤더가 스왑 공간 계산에 대한 권장 사항을 제공하는 경우도 있습니다. 다음 표에는 실제 메모리 총량에 따른 지침이 나와 있습니다.

RAM 및 스왑 공간 권장 사항

RAM	스왑 공간	최대 절전 모드를 허용하는 경우 스왑 공간
2GB 이하	RAM의 두 배	RAM의 3배
2~8GB	RAM과 동일	RAM의 두 배
8~64GB	4GB 이상	RAM의 1.5배
64GB 초과	4GB 이상	최대 절전 모드는 권장되지 않습니다.

랩탑 및 데스크탑 최대 절전 모드 기능은 시스템의 전원을 끄기 전에 스왑 공간을 사용하여 RAM 내용을 저장합니다. 시스템을 다시 켜면 커널이 스왑 공간에서 RAM 내용을 복원하므로 완전한 부팅이 필요하지 않습니다. 해당 시스템의 경우 스왑 공간이 RAM 크기보다 커야 합니다.

이 섹션의 끝에 있는 참조의 기술 자료 문서에서 스왑 공간 크기 조정에 대한 추가 지침을 제공합니다.

스왑 공간 생성

스왑 공간을 생성하려면 다음 단계를 수행합니다.

9장 | 기본 스토리지 관리

- 파일 시스템 유형이 **linux-swap**인 파티션을 생성합니다.
- 장치에서 스왑 시그니처를 저장합니다.

스왑 파티션 생성

parted 명령을 사용하여 원하는 크기의 파티션을 생성하고 해당 파일 시스템 유형을 **linux-swap**으로 설정합니다. 이전의 툴은 파티션 파일 시스템 유형으로 장치의 활성화 여부를 결정했지만 이제는 아닙니다. 유틸리티에서는 더 이상 파티션 파일 시스템 유형을 사용하지 않지만 관리자는 해당 유형을 통해 파티션의 용도를 확인할 수 있습니다.

다음 예제에서는 256MB 파티션을 생성합니다.

```
[root@host ~]# parted /dev/vdb
GNU Parted 3.4
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB          data

(parted) mkpart
Partition name?  []? swap1
File system type? [ext2]? linux-swap
Start? 1001MB
End? 1257MB
(parted) print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB          data
 2      1001MB  1257MB  256MB   linux-swap(v1)  swap1

(parted) quit
Information: You may need to update /etc/fstab.

[root@host ~]#
```

파티션을 생성한 후 **udevadm settle** 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 감지하고 **/dev** 디렉터리에 관련 장치 파일을 생성할 때까지 기다립니다. 해당 명령은 완료될 때만 반환됩니다.

```
[root@host ~]# udevadm settle
```

스왑 공간 포맷

mkswap 명령은 스왑 시그니처를 장치에 적용합니다. 다른 포맷 유ти리티와 달리 **mkswap** 명령은 장치의 시작 부분에 단일 데이터 블록을 쓰고 장치의 나머지 부분은 커널에서 메모리 페이지를 저장하는 데 사용할 수 있도록 포맷되지 않은 상태로 둡니다.

```
[root@host ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 244 MiB (255848448 bytes)
no label, UUID=39e2667a-9458-42fe-9665-c5c854605881
```

스왑 공간 활성화

swapon 명령을 사용하여 포맷된 스왑 공간을 활성화할 수 있습니다.

장치를 매개 변수로 하여 **swapon**을 사용하거나 **swapon -a**를 사용하여 **/etc/fstab** 파일에 나열된 모든 스왑 공간을 활성화합니다. **swapon --show** 및 **free** 명령을 사용하여 사용 가능한 스왑 공간을 검사합니다.

```
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:       1873036       134688      1536436          16748      201912       1576044
Swap:            0           0           0
[root@host ~]# swapon /dev/vdb2
[root@host ~]# free
              total        used        free      shared  buff/cache   available
Mem:       1873036       135044      1536040          16748      201952       1575680
Swap:       249852           0      249852
```

swapoff 명령을 사용하여 스왑 공간을 비활성화할 수 있습니다. 페이지가 스왑 공간에 작성된 경우 **swapoff** 명령은 해당 페이지를 다른 활성 스왑 공간으로 이동하거나 메모리로 다시 이동하려고 합니다. **swapoff** 명령에서 데이터를 다른 위치에 쓸 수 없는 경우 명령이 오류와 함께 실패하고, 스왑 공간은 활성 상태로 유지됩니다.

스왑 공간 영구 활성화

/etc/fstab 파일에 항목을 생성하여 시스템 부팅 시 활성 스왑 공간을 확보합니다. 아래 예제에서는 위에서 생성한 스왑 공간을 기준으로 **/etc/fstab**의 일반적인 행을 보여줍니다.

```
UUID=39e2667a-9458-42fe-9665-c5c854605881    swap    swap    defaults    0 0
```

예에서는 첫 번째 필드로 UUID를 사용합니다. 장치를 포맷할 때 **mkswap** 명령은 해당 UUID를 표시합니다. **mkswap**의 출력이 손실된 경우 **lsblk --fs** 명령을 사용하십시오. 또는 첫 번째 필드에 장치 이름을 사용해도 됩니다.

두 번째 필드는 일반적으로 마운트 지점에 예약되어 있습니다. 하지만 디렉터리 구조를 통해 액세스할 수 없는 스왑 장치의 경우 이 필드는 자리 표시자 값인 **swap**을 사용합니다. **fstab(5)** 도움말 페이지에서는 자리 표시자 값 **none**을 사용하지만, 값 **swap**을 사용하면 문제가 발생하는 경우 자세한 정보를 제공하는 오류 메시지가 표시됩니다.

세 번째 필드는 파일 시스템 유형입니다. 스왑 공간의 파일 시스템 유형은 **swap**입니다.

네 번째 필드는 옵션용입니다. 이 예제에서는 **defaults** 옵션을 사용합니다. **defaults** 옵션에는 시스템 부팅 시 스왑 공간을 자동으로 활성화하는 **auto** 마운트 옵션이 포함되어 있습니다.

마지막 두 필드는 **dump** 플래그와 **fsck** 순서입니다. 스왑 공간은 백업 또는 파일 시스템 점검이 필요하지 않으므로 해당 필드를 0으로 설정해야 합니다.

/etc/fstab 파일에 항목을 추가하거나 제거할 때 **systemctl daemon-reload** 명령을 실행하거나, **systemd**가 새 구성을 등록하도록 서버를 재부팅합니다.

```
[root@host ~]# systemctl daemon-reload
```

스왑 공간 우선순위 설정

기본적으로 시스템은 스왑 공간을 직렬로 사용합니다. 즉, 커널은 첫 번째로 활성화된 스왑 공간을 가득 찰 때 까지 사용한 다음 두 번째 스왑 공간을 사용하기 시작합니다. 그러나 각 스왑 공간에 우선순위를 정의하여 특정 순서를 강제로 적용할 수 있습니다.

우선순위를 설정하려면 **/etc/fstab** 파일에서 **pri** 옵션을 사용합니다. 커널은 우선 순위가 가장 높은 스왑 공간을 먼저 사용합니다. 기본 우선 순위는 -2입니다.

다음 예제에서는 **/etc/fstab** 파일에 정의된 세 가지 스왑 공간을 보여줍니다. 커널은 우선순위가 10으로 설정되어 있으므로 마지막 항목을 먼저 사용합니다. 해당 공간이 가득 차면 우선순위가 4로 설정되어 있는 두 번째 항목을 사용합니다. 마지막으로 기본 우선순위가 -2인 첫 번째 항목을 사용합니다.

```
UUID=af30cbb0-3866-466a-825a-58889a49ef33    swap    swap    defaults  0 0
UUID=39e2667a-9458-42fe-9665-c5c854605881    swap    swap    pri=4      0 0
UUID=fb7fa60-b781-44a8-961b-37ac3ef572bf     swap    swap    pri=10     0 0
```

스왑 공간 우선순위를 표시하려면 **swapon --show** 명령을 사용합니다.

스왑 공간의 우선 순위가 같을 경우 커널은 라운드 로빈 방식으로 스왑 공간에 씁니다.



참조

mkswap(8), **swapon(8)**, **swapoff(8)**, **mount(8)** 및 **parted(8)** 도움말 페이지

지식베이스: Red Hat 플랫폼에 권장되는 스왑 크기는 무엇입니까?

<https://access.redhat.com/solutions/15244>

▶ 연습 가이드

스왑 공간 관리

이 연습에서는 스왑 공간으로 사용할 파티션을 생성하고 포맷한 다음 영구적으로 활성화합니다.

결과

- GPT 파티셔닝 체계를 사용하여 디스크에 파티션과 스왑 공간을 생성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start storage-swap
```

지침

- ▶ 1. **servera**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **/dev/vdb** 디스크를 검사합니다. 디스크에 이미 파티션 테이블이 있고 GPT 파티셔닝 체계를 사용합니다. 또한 기존의 1GB 파티션이 있습니다.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  1001MB  1000MB          data
```

- ▶ 3. 스왑 공간으로 사용할 500MB의 새 파티션을 추가합니다. 파티션 유형을 **linux-swap**으로 설정합니다.

- 3.1. **myswap** 파티션을 생성합니다. 디스크에서 GPT 파티셔닝 체계를 사용하기 때문에 파티션에 이름을 지정해야 합니다. 시작 위치인 1001MB는 기존 첫 번째 파티션의 끝입니다. **parted** 명령을 사용하면 새 파티션이 간격 없이 이전 파티션 바로 뒤에 이어집니다. 파티션이 1001MB 위

9장 | 기본 스토리지 관리

치에서 시작하기 때문에 이 명령은 500MB의 파티션 크기를 얻기 위해 끝 위치를 1501MB로 설정합니다.

```
[root@servera ~]# parted /dev/vdb mkpart myswap linux-swap \
1001MB 1501MB
Information: You may need to update /etc/fstab.
```

- 3.2. **/dev/vdb** 디스크의 파티션을 나열하여 작업을 확인합니다. 새 파티션의 크기는 정확히 500MB가 아닙니다. 이러한 크기 차이는 **parted** 명령이 파티션을 디스크 레이아웃에 맞춰야 하기 때문에 발생합니다.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name   Flags
 1      1049kB  1001MB  999MB   data
 2      1001MB  1501MB  499MB   myswap  swap
```

- 3.3. **udevadm settle** 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 등록할 때까지 기다린 후 완료되면 반환됩니다.

```
[root@servera ~]# udevadm settle
```

▶ 4. 새 파티션을 스왑 공간으로 초기화합니다.

```
[root@servera ~]# mkswap /dev/vdb2
Setting up swapspace version 1, size = 476 MiB (499118080 bytes)
no label, UUID=cb7f71ca-ee82-430e-ad4b-7dda12632328
```

▶ 5. 새 스왑 공간을 활성화합니다.

- 5.1. 스왑 공간을 생성하고 초기화해도 아직 활성화되지는 않은 것을 확인합니다.

```
[root@servera ~]# swapon --show
```

- 5.2. 새 스왑 공간을 활성화합니다.

```
[root@servera ~]# swapon /dev/vdb2
```

- 5.3. 새 스왑 공간을 이제 사용할 수 있는지 확인합니다.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

- 5.4. 스왑 공간을 비활성화합니다.

```
[root@servera ~]# swapoff /dev/vdb2
```

5.5. 스왑 공간이 비활성화되었는지 확인합니다.

```
[root@servera ~]# swapon --show
```

▶ 6. 시스템 부팅 시 새 스왑 공간을 활성화합니다.

- 6.1. **lsblk** 명령에 **--fs** 옵션을 사용하여 **/dev/vdb2** 장치의 UUID를 검색합니다. 출력의 UUID는 시스템마다 다릅니다.

```
[root@servera ~]# lsblk --fs /dev/vdb2
NAME FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
MOUNTPOINTS
vdb2 swap    1          762735cb-a52a-4345-9ed0-e3a68aa8bb97
```

- 6.2. **/etc/fstab** 파일에 다음 항목을 추가합니다. 다음 명령의 UUID를 위 단계에서 검색한 UUID로 바꿉니다.

```
...output omitted...
UUID=762735cb-a52a-4345-9ed0-e3a68aa8bb97  swap  swap  defaults  0 0
```

- 6.3. 시스템의 **systemd** 데몬을 업데이트하여 새 **/etc/fstab** 파일 구성을 등록합니다.

```
[root@servera ~]# systemctl daemon-reload
```

- 6.4. **/etc/fstab** 파일의 항목을 사용하여 스왑 공간을 활성화합니다.

```
[root@servera ~]# swapon -a
```

- 6.5. 새 스왑 공간이 활성화되었는지 확인합니다.

```
[root@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B   -2
```

▶ 7. **servera** 시스템을 재부팅합니다. 서버가 재부팅되면 로그인하여 스왑 공간이 활성화되었는지 확인합니다. 완료되면 **servera**에서 로그아웃합니다.

- 7.1. **servera** 시스템을 재부팅합니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- 7.2. **servera**가 재부팅될 때까지 기다린 후 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

7.3. 스왑 공간이 활성화되었는지 확인합니다.

```
[student@servera ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 476M   0B    -2
```

7.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish storage-swap
```

이것으로 섹션을 완료합니다.

▶ 랩

기본 스토리지 관리

이 랩에서는 새 디스크에 여러 개의 파티션을 생성한 뒤 일부 파티션은 파일 시스템으로 포맷한 다음 마운트하고 나머지는 스왑 공간으로 활성화합니다.

결과

- **parted** 명령을 사용하여 파티션을 표시하고 생성합니다.
- 파티션에 새 파일 시스템을 생성하고 영구적으로 마운트합니다.
- 스왑 공간을 생성하고 부팅 시 활성화합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start storage-review
```

지침

1. **serverb** 시스템에 사용되지 않은 디스크가 여러 개 있습니다. 사용되지 않은 첫 번째 새 디스크에서 GPT 파티션 레이블과 **backup**이라는 2GB GPT 파티션을 생성합니다.
정확한 크기를 설정하기가 어려우므로 1.8~2.2GB의 크기가 허용됩니다.
XFS 파일 시스템을 호스팅하도록 **backup** 파티션을 구성합니다.
2. 2GB **backup** 파티션을 XFS 파일 시스템으로 포맷하고 **/backup** 디렉터리에 영구적으로 마운트합니다.
3. 동일한 디스크에 **swap1** 및 **swap2**라는 512MB GPT 파티션 두 개를 생성합니다.
460~564MB의 크기가 허용됩니다.
스왑 공간을 호스트하도록 파티션의 파일 시스템 유형을 구성합니다.
4. 두 개의 512MB 파티션을 스왑 공간으로 초기화하고 부팅 시 활성화되도록 구성합니다. **swap2** 파티션의 스왑 공간이 다른 파티션보다 우선 적용되도록 설정합니다. 512MB는 약 488MiB에 해당합니다.
5. 작업을 확인하려면 **serverb** 시스템을 재부팅합니다. 시스템이 첫 번째 파티션을 **/backup** 디렉터리에 자동으로 마운트하는지 확인합니다. 또한 시스템이 두 스왑 공간을 활성화하는지 확인합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade storage-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish storage-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

기본 스토리지 관리

이 랩에서는 새 디스크에 여러 개의 파티션을 생성한 뒤 일부 파티션은 파일 시스템으로 포맷한 다음 마운트하고 나머지는 스왑 공간으로 활성화합니다.

결과

- **parted** 명령을 사용하여 파티션을 표시하고 생성합니다.
- 파티션에 새 파일 시스템을 생성하고 영구적으로 마운트합니다.
- 스왑 공간을 생성하고 부팅 시 활성화합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start storage-review
```

지침

1. **serverb** 시스템에 사용되지 않은 디스크가 여러 개 있습니다. 사용되지 않은 첫 번째 새 디스크에서 GPT 파티션 레이블과 **backup**이라는 2GB GPT 파티션을 생성합니다.
정확한 크기를 설정하기가 어려우므로 1.8~2.2GB의 크기가 허용됩니다.
XFS 파일 시스템을 호스팅하도록 **backup** 파티션을 구성합니다.

- 1.1. **serverb**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. 사용되지 않은 디스크를 확인합니다. 사용되지 않은 첫 번째 디스크인 **/dev/vdb**에 파티션이 없습니다.

```
[root@serverb ~]# lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
vda    252:0    0   10G  0 disk
└─vda1 252:1    0    1M  0 part
└─vda2 252:2    0  200M  0 part /boot/efi
└─vda3 252:3    0  500M  0 part /boot
└─vda4 252:4    0  9.3G  0 part /
```

```
vdb      252:16   0    5G  0 disk
vdc      252:32   0    5G  0 disk
vdd      252:48   0    5G  0 disk
```

1.3. `/dev/vdb` 디스크에 레이블이 없는지 확인합니다.

```
[root@serverb ~]# parted /dev/vdb print
Error: /dev/vdb: unrecognised disk label
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
```

1.4. GPT 파티셔닝 체계를 정의합니다.

```
[root@serverb ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.
```

1.5. 파일 시스템 유형이 `xfs`인 2GB `backup` 파티션을 생성합니다. 섹터 2048에서 파티션을 시작합니다.

```
[root@serverb ~]# parted /dev/vdb mkpart backup xfs 2048s 2GB
Information: You may need to update /etc/fstab.
```

1.6. `backup` 파티션이 생성되었는지 확인합니다.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start    End     Size   File system  Name     Flags
 1       1049kB  2000MB  1999MB        xfs        backup
```

1.7. `udevadm settle` 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 감지하고 `/dev/vdb1` 장치 파일을 생성할 때까지 기다립니다.

```
[root@serverb ~]# udevadm settle
```

2. 2GB `backup` 파티션을 XFS 파일 시스템으로 포맷하고 `/backup` 디렉터리에 영구적으로 마운트합니다.

2.1. `/dev/vdb1` 파티션을 포맷합니다.

```
[root@serverb ~]# mkfs.xfs /dev/vdb1
meta-data=/dev/vdb1              isize=512    agcount=4, agsize=121984 blks
                                =                      sectsz=512  attr=2, projid32bit=1
                                =                      crc=1      finobt=1, sparse=1, rmapbt=0
```

```

        =           reflink=1    bigtime=1 inobtcount=1
data   =           bsize=4096   blocks=487936, imaxpct=25
        =           sunit=0     swidth=0 blks
naming =version 2  bsize=4096   ascii-ci=0, ftype=1
log    =internal log bsize=4096   blocks=2560, version=2
        =           sectsz=512  sunit=0 blks, lazy-count=1
realtime =none      extsz=4096  blocks=0, rtextents=0

```

- 2.2. `/backup` 마운트 지점을 만듭니다.

```
[root@serverb ~]# mkdir /backup
```

- 2.3. `/etc/fstab` 파일에 새 파일 시스템을 추가하기 전에 해당 UUID를 검색합니다. 해당 시스템의 UUID는 다를 수 있습니다.

```
[root@serverb ~]# lsblk --fs /dev/vdb1
NAME FSTYPE FSVER LABEL UUID                                     FSAVAIL FSUSE%
MOUNTPOINTS
vdb1 xfs          f74ed805-b1fc-401a-a5ee-140f97c6757d
```

- 2.4. `/etc/fstab` 파일을 편집하여 새 파일 시스템을 정의합니다.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=f74ed805-b1fc-401a-a5ee-140f97c6757d  /backup  xfs  defaults  0 0
```

- 2.5. `systemd` 데몬이 `/etc/fstab` 파일을 다시 읽도록 강제 실행합니다.

```
[root@serverb ~]# systemctl daemon-reload
```

- 2.6. `/backup` 디렉터리를 수동으로 마운트하여 작업을 확인합니다. 마운트에 성공했는지 확인합니다.

```
[root@serverb ~]# mount /backup
[root@serverb ~]# mount | grep /backup
/dev/vdb1 on /backup type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
```

3. 동일한 디스크에 `swap1` 및 `swap2`라는 512MB GPT 파티션 두 개를 생성합니다.

460~564MB의 크기가 허용됩니다.

스왑 공간을 호스트하도록 파티션의 파일 시스템 유형을 구성합니다.

- 3.1. `/dev/vdb` 디스크에서 현재 파티션 테이블을 표시하여 첫 번째 파티션의 끝 위치를 검색합니다. 다음 단계에서 해당 값을 `swap1` 파티션의 시작으로 사용합니다.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
```

```
Disk Flags:
```

Number	Start	End	Size	File system	Name	Flags
1	1049kB	2000MB	1999MB	xfs		backup

- 3.2. swap1이라는 첫 번째 512MB GPT 파티션을 생성합니다. 유형을 linux-swap으로 설정합니다. 첫 번째 파티션의 끝 위치를 시작점으로 사용합니다. 끝 위치는 2000MB + 512MB = 2512MB입니다.

```
[root@serverb ~]# parted /dev/vdb mkpart swap1 linux-swap 2000M 2512M
Information: You may need to update /etc/fstab.
```

- 3.3. swap2라는 두 번째 512MB GPT 파티션을 생성합니다. 유형을 linux-swap으로 설정합니다. 이전 파티션의 끝 위치인 2512M을 시작점으로 사용합니다. 끝 위치는 2512MB + 512MB = 3024MB입니다.

```
[root@serverb ~]# parted /dev/vdb mkpart swap2 linux-swap 2512M 3024M
Information: You may need to update /etc/fstab.
```

- 3.4. 파티션 테이블을 표시하여 작업을 확인합니다.

```
[root@serverb ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name     Flags
 1      1049kB  2000MB  1999MB  xfs        backup
 2      2000MB  2512MB  513MB   swap1      swap
 3      2512MB  3024MB  512MB   swap2      swap
```

- 3.5. udevadm settle 명령을 실행합니다. 이 명령은 시스템이 새 파티션을 등록하고 장치 파일을 생성할 때까지 기다립니다.

```
[root@serverb ~]# udevadm settle
```

4. 두 개의 512MB 파티션을 스왑 공간으로 초기화하고 부팅 시 활성화되도록 구성합니다. swap2 파티션의 스왑 공간이 다른 파티션보다 우선 적용되도록 설정합니다. 512MB는 약 488MiB에 해당합니다.

- 4.1. mkswap 명령을 사용하여 스왑 파티션을 초기화합니다. 다음 단계에서 해당 정보를 사용하므로 두 스왑 공간의 UUID를 확인합니다. mkswap 출력을 지우면 lsblk --fs 명령을 사용하여 UUID를 검색합니다.

```
[root@serverb ~]# mkswap /dev/vdb2
Setting up swap space version 1, size = 489 MiB (512749568 bytes)
no label, UUID=87976166-4697-47b7-86d1-73a02f0fc803
[root@serverb ~]# mkswap /dev/vdb3
Setting up swap space version 1, size = 488 MiB (511700992 bytes)
no label, UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942
```

- 4.2. `/etc/fstab` 파일을 편집하여 새 스왑 공간을 정의합니다. `swap1` 파티션보다 우선 적용되도록 `swap2` 파티션의 스왑 공간을 설정하려면 `pri` 옵션을 사용하여 `swap2` 파티션에 더 높은 우선순위를 지정합니다.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
UUID=a3665c6b-4fb-49b6-a528-74e268b058dd  /backup xfs defaults 0 0
UUID=87976166-4697-47b7-86d1-73a02f0fc803 swap swap pri=10 0 0
UUID=4d9b847b-98e0-4d4e-9ef7-dfaaf736b942 swap swap pri=20 0 0
```

- 4.3. `systemd` 데몬이 `/etc/fstab` 파일을 다시 읽도록 강제 실행합니다.

```
[root@serverb ~]# systemctl daemon-reload
```

- 4.4. 새 스왑 공간을 활성화합니다. 스왑 공간이 올바르게 활성화되었는지 확인합니다.

```
[root@serverb ~]# swapon -a
[root@serverb ~]# swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

5. 작업을 확인하려면 `serverb` 시스템을 재부팅합니다. 시스템이 첫 번째 파티션을 `/backup` 디렉터리에 자동으로 마운트하는지 확인합니다. 또한 시스템이 두 스왑 공간을 활성화하는지 확인합니다.

- 5.1. `serverb`을 재부팅합니다.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

- 5.2. `serverb` 가 부팅될 때까지 기다린 후 `student` 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 5.3. 시스템이 `/dev/vdb1` 파티션을 `/backup` 디렉터리에 자동으로 마운트하는지 확인합니다.

```
[student@serverb ~]$ mount | grep /backup
/dev/vdb1 on /backup type xfs (rw,relatime,seclabel,attr2,inode64,noquota)
```

- 5.4. 시스템이 두 스왑 공간을 모두 활성화하는지 확인합니다.

```
[student@serverb ~]$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/vdb2 partition 489M   0B   10
/dev/vdb3 partition 488M   0B   20
```

5.5. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.  
[student@workstation ~]$
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade storage-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish storage-review
```

이것으로 섹션을 완료합니다.

요약

- **root** 사용자는 **mount** 명령을 사용하여 파일 시스템을 수동으로 마운트할 수 있습니다.
- 장치를 성공적으로 마운트 해제하려면 모든 프로세스에서 마운트 지점에 대한 액세스를 중지해야 합니다.
- 그래픽 환경을 사용하는 경우 이동식 스토리지 장치는 **/run/media** 디렉터리에 마운트됩니다.
- **lsblk** 명령은 크기 및 UUID와 같은 블록 장치 세부 정보를 표시합니다.
- **parted** 명령은 MBR 또는 GPT 파티셔닝 체계를 사용하여 디스크에서 파티션을 추가, 수정, 제거합니다.
- **mkfs.xfs** 명령은 디스크 파티션에 XFS 파일 시스템을 생성합니다.
- **/etc/fstab** 파일에는 영구적으로 마운트해야 하는 장치가 포함되어 있습니다.
- **mkswap** 명령은 스왑 공간을 초기화합니다.

스토리지 스택 관리

목적

명령줄에서 파일 시스템 또는 스왑 공간이 포함된 논리 볼륨을 생성 및 관리합니다.

목표

- 논리 볼륨 관리자 구성 요소 및 개념을 설명하고, LVM 스토리지를 구현하고, LVM 구성 요소 정보를 표시합니다.

섹션

- 논리 볼륨 생성 및 확장(안내에 따른 연습)

랩

- 스토리지 스택 관리

논리 볼륨 생성 및 확장

목표

논리 볼륨 관리자 구성 요소 및 개념을 설명하고, LVM 스토리지를 구현하고, LVM 구성 요소 정보를 표시합니다.

논리 볼륨 관리자 개요

LVM(논리 볼륨 관리자) 시스템을 사용하여 논리 스토리지 볼륨을 실제 스토리지에 계층으로 생성합니다. 이 스토리지 시스템은 실제 스토리지를 직접 사용하는 것보다 더 큰 유연성을 제공합니다. LVM을 사용하면 소프트웨어에서 하드웨어 스토리지 구성을 숨기고, 애플리케이션을 중지하거나 파일 시스템을 마운트 해제하지 않고도 볼륨 크기를 조정할 수 있습니다. LVM은 스토리지를 관리하는 포괄적인 명령줄 도구를 제공합니다.

물리 장치

논리 볼륨은 데이터를 저장하는 데 물리 장치를 사용합니다. 이러한 장치는 디스크 파티션, 전체 디스크, RAID 어레이 또는 SAN 디스크일 수 있습니다. 장치를 LVM 물리 볼륨으로 초기화해야 합니다. 하나의 LVM 물리 볼륨이 전체 물리 장치를 사용해야 합니다.

PV(물리 볼륨)

LVM은 기본 물리 장치를 LVM 물리 볼륨으로 사용합니다. LVM 툴은 물리 볼륨을 PE(물리 확장)로 분할하여 PV에서 가장 작은 스토리지 블록 역할을 하는 작은 데이터 청크를 형성합니다.

VG(볼륨 그룹)

볼륨 그룹은 하나 이상의 PV로 구성된 스토리지 풀입니다. 실제 스토리지와 기능적으로 전체 디스크와 동일합니다. 하나의 PV는 단일 VG에만 할당할 수 있습니다. LVM은 PE 크기를 자동으로 설정하며, 지정 할 수도 있습니다. VG는 사용되지 않은 공간과 여러 논리 볼륨으로 구성될 수 있습니다.

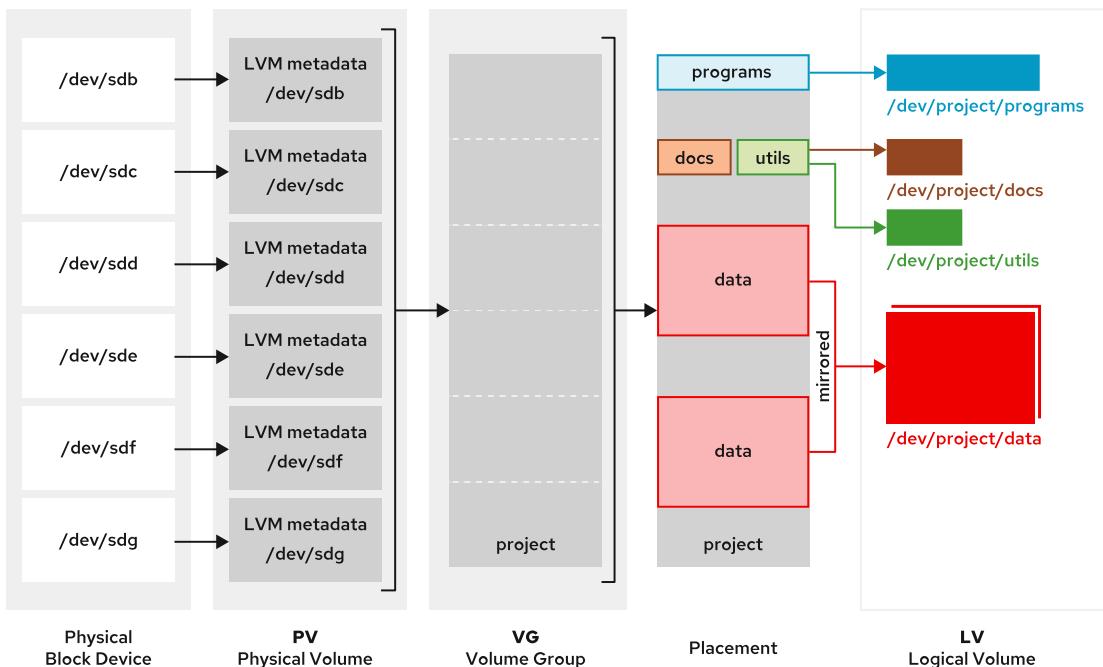
LV(논리 볼륨)

논리 볼륨은 VG의 사용하지 않는 물리 확장 영역에서 생성되며 애플리케이션, 사용자, 운영 체제를 위한 스토리지 장치로 제공됩니다. LV는 물리 확장 영역에 매핑되는 LE(논리 확장 영역)의 컬렉션입니다. 기본적으로 각 LE는 하나의 PE에 매핑됩니다. 특정 LV 옵션을 설정하면 이 매핑이 변경됩니다. 예를 들어, 미러링의 경우 각 LE가 두 개의 PE에 매핑됩니다.

논리 볼륨 관리자 워크플로

LVM 스토리지를 생성하려면 논리적 워크플로에서 구조를 구축해야 합니다.

- 물리 볼륨을 생성하는 데 사용되는 물리 장치를 확인하고 이러한 장치를 LVM 물리 볼륨으로 초기화합니다.
- 여러 물리 볼륨에서 볼륨 그룹을 생성합니다.
- 볼륨 그룹의 사용 가능한 공간에서 논리 볼륨을 생성합니다.
- 파일 시스템으로 논리 볼륨을 포맷하고 마운트하거나, 스왑 공간으로 활성화하거나, 고급 구조를 위해 원시 볼륨을 데이터베이스 또는 스토리지 서버에 전달합니다.

**참고**

이 예제에서는 `/dev/vdb` 장치 이름과 해당 스토리지 파티션을 사용합니다. 강의실 시스템의 장치 이름은 다를 수 있습니다. 해당 시스템의 장치를 확인하려면 `lsblk`, `blkid` 또는 `cat /proc/partitions` 명령을 사용합니다.

LVM 스토리지 구축

논리 볼륨을 생성하려면 물리 장치 파티션, 물리 볼륨, 볼륨 그룹을 생성해야 합니다. LV를 생성한 후 볼륨을 포맷하고 마운트하여 스토리지로 액세스합니다.

물리 장치 준비

파티션이 이미 있는 경우 파티셔닝은 선택 사항입니다. 물리 장치에 파티션을 생성하려면 `parted` 명령을 사용합니다. 물리 장치를 Linux LVM 파티션 유형으로 설정합니다. 커널에 새 파티션을 등록하려면 `udevadm settle` 명령을 사용합니다.

```
[root@host ~]# parted /dev/vdb mklabel gpt mkpart primary 1MiB 769MiB
...output omitted...
[root@host ~]# parted /dev/vdb mkpart primary 770MiB 1026MiB
[root@host ~]# parted /dev/vdb set 1 lvm on
[root@host ~]# parted /dev/vdb set 2 lvm on
[root@host ~]# udevadm settle
```

물리 볼륨 생성

`pvccreate` 명령을 사용하여 물리 파티션의 레이블을 LVM 물리 볼륨으로 지정합니다. `pvccreate` 명령에 공백으로 구분된 장치 이름을 인수로 사용하여 여러 개의 장치에 동시에 레이블을 지정할 수 있습니다. 이 예제에서는 `/dev/vdb1` 및 `/dev/vdb2` 장치에 볼륨 그룹을 생성할 준비가 된 PV라는 레이블을 지정합니다.

```
[root@host ~]# pvcreate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created.
Physical volume "/dev/vdb2" successfully created.
Creating devices file /etc/lvm/devices/system.devices
```

볼륨 그룹 생성

vgcreate 명령은 하나 이상의 물리 볼륨을 볼륨 그룹으로 빌드합니다. 첫 번째 인수는 볼륨 그룹 이름이며 이 VG에 할당할 하나 이상의 물리 볼륨이 뒤에 옵니다. 이 예제에서는 **/dev/vdb1** 및 **/dev/vdb2** PV를 사용하여 **vg01** VG를 생성합니다.

```
[root@host ~]# vgcreate vg01 /dev/vdb1 /dev/vdb2
Volume group "vg01" successfully created
```

논리 볼륨 생성

lvcreate 명령은 볼륨 그룹의 사용 가능한 PE에서 논리 볼륨을 생성합니다. **lvcreate** 명령을 사용하여 이 논리 볼륨을 포함할 LV 이름과 크기 및 VG 이름을 설정합니다. 이 예제에서는 **vg01** VG에 300MiB **lv01** LV를 생성합니다.

```
[root@host ~]# lvcreate -n lv01 -L 300M vg01
Logical volume "lv01" created.
```

이 명령은 볼륨 그룹에 사용 가능한 물리 확장 영역의 수가 충분할 때만 성공합니다. 크기가 정확히 일치하지 않는 경우 LV 크기는 다음번 PE 크기 값으로 반올림됩니다.

lvcreate 명령 **-L** 옵션에는 바이트, 메비바이트(메가 이진 바이트, 1048576바이트), 기비바이트(기가 이진 바이트) 등으로 된 크기를 사용해야 합니다. 소문자 **-l**에는 물리 확장 영역의 개수로 지정된 크기가 필요합니다. 다음은 동일한 크기의 동일한 LV를 생성하는 두 가지 선택 가능한 명령입니다.

- **lvcreate -n lv01 -L 128M vg01** : 다음 PE로 반올림된 128MiB 크기의 LV를 만듭니다.
- **lvcreate -n lv01 -l 32 vg01** : 총 128MiB이며 각 4MiB인 크기가 32PE의 LV를 만듭니다.

중복 제거 및 압축을 사용하여 논리 볼륨 생성

RHEL 9에서는 LVM VDO 구현을 사용하여 VDO 볼륨을 관리합니다. 이전 Python 기반 VDO 관리 툴을 계속 사용할 수 있지만 더 이상 필요하지는 않습니다.

VDO(가상 데이터 최적화 도구)는 스토리지에 인라인 블록 수준의 중복 제거, 압축, 씬 프로비저닝 기능을 제공합니다. 최대 256TB의 실제 스토리지를 사용하도록 VDO 볼륨을 구성합니다. LVM 씬 프로비저닝 볼륨과 유사한 LVM LV(논리 볼륨) 유형으로 VDO를 관리합니다. LVM VDO는 두 개의 논리 볼륨으로 구성됩니다.

VDO 풀 LV

이 LV는 데이터의 저장, 중복 제거, 압축을 수행하고 물리 장치에서 지원하는 VDO 볼륨의 크기를 설정합니다. 각 VDO 풀 LV는 하나의 VDO LV만 보유할 수 있으므로 VDO는 중복 제거되고 각 VDO LV를 개별적으로 압축합니다.

VDO LV

가상 장치는 중복 제거 및 압축이 발생하기 전에 VDO 풀 LV 위에 프로비저닝되고 데이터를 저장하는 VDO 볼륨의 논리적 크기를 설정합니다.

LVM VDO는 중복 제거된 스토리지를 일반 LV(논리 볼륨)로 제공합니다. VDO 볼륨은 표준 파일 시스템으로 포맷하거나, 블록 장치로 공유하거나, 일반 논리 볼륨과 마찬가지로 다른 스토리지 계층을 빌드하는 데 사용할 수 있습니다.

VDO 중복 제거 및 압축을 사용하려면 **vdo** 및 **kmod-kvdo** 패키지를 설치하십시오.

```
[root@host ~]# dnf install vdo kmod-kvdo
```

선택한 LVM 볼륨 그룹에 사용 가능한 스토리지 용량이 충분한지 확인합니다. **lvcreate** 명령을 **--type vdo** 매개 변수와 함께 사용하여 VDO LV를 생성합니다.

```
[root@host ~]# lvcreate --type vdo --name vdo-lv01 --size 5G vg01
Logical blocks defaulted to 523108 blocks.
The VDO volume can address 2 GB in 1 data slab.
It can grow to address at most 16 TB of physical storage in 8192 slabs.
If a larger maximum size might be needed, use bigger slabs.
Logical volume "vdo-lv01" created.
```

논리 볼륨에 파일 시스템 생성

기존 이름(**/dev/vgname/lvname**) 또는 커널 장치 매퍼 이름(**/dev/mapper/vgname-lvname**)을 사용하여 논리 볼륨을 표시합니다.

mkfs 명령을 사용하여 새 논리 볼륨에 파일 시스템을 생성합니다.

```
[root@host ~]# mkfs -t xfs /dev/vg01/lv01
...output omitted...
```

mkdir 명령을 사용하여 마운트 지점을 생성합니다.

```
[root@host ~]# mkdir /mnt/data
```

파일 시스템을 영구적으로 사용할 수 있도록 하려면 **/etc/fstab** 파일에 항목을 추가합니다.

```
/dev/vg01/lv01 /mnt/data xfs defaults 0 0
```

mount 명령을 사용하여 LV를 마운트합니다.

```
[root@host ~]# mount /mnt/data/
```



참고

LVM은 UUID를 찾는 PV를 구문 분석하므로 이름 또는 UUID로 논리 볼륨을 마운트할 수 있습니다. 이 동작은 PV에 항상 UUID가 포함되므로 이름을 사용하여 VG를 생성한 경우에도 성공합니다.

LVM 구성 요소 상태 표시

LVM은 PV, VG, LV의 상태 정보를 표시하는 다양한 유틸리티를 제공합니다. LVM 구성 요소의 상태 정보를 표시하려면 **pvs**, **vgs**, **lvs** 명령이 일반적으로 사용되며 상태 정보의 하위 집합이 엔터티당 한 행으로 표시됩니다.

연관된 **pvs**, **vgs**, **lvs** 명령이 일반적으로 사용되며 상태 정보의 하위 집합이 엔터티당 한 행으로 표시됩니다.

물리 볼륨 정보 표시

pvdisplay 명령은 PV에 대한 정보를 표시합니다. 이 명령을 인수 없이 사용하면 시스템에 있는 모든 PV의 정보가 나열됩니다. PV 이름을 명령에 인수로 제공하면 해당 PV와 관련된 정보가 표시됩니다.

```
[root@host ~]# pvdisplay /dev/vdb1
--- Physical volume ---
PV Name          /dev/vdb1          ①
VG Name          vg01              ②
PV Size          731.98 MiB / not usable 3.98 MiB ③
Allocatable      yes
PE Size          4.00 MiB          ④
Total PE         182
Free PE          107              ⑤
Allocated PE     75
PV UUID          zP0gD9-NxTV-Qtoi-yfQD-TGpL-0Yj0-wExh2N
```

- ① PV Name에는 장치 이름이 표시됩니다.
- ② VG Name에는 PV가 할당된 볼륨 그룹이 표시됩니다.
- ③ PV Size에는 사용할 수 없는 공간을 포함하여 PV의 물리적 크기가 표시됩니다.
- ④ PE Size에는 물리 확장 영역의 크기가 표시됩니다.
- ⑤ Free PE에는 VG에서 새 LV를 생성하거나 기존 LV를 확장하는 데 사용할 수 있는 PE 크기가 표시됩니다.

볼륨 그룹 정보 표시

vgdisplay 명령은 볼륨 그룹에 대한 정보를 표시합니다. 모든 VG에 대한 정보를 나열하려면 이 명령을 인수 없이 사용하십시오. VG 이름을 인수로 제공하면 해당 VG와 관련된 정보가 나열됩니다.

```
[root@host ~]# vgdisplay vg01
--- Volume group ---
VG Name          vg01          ①
System ID
Format           lvm2
Metadata Areas   2
Metadata Sequence No  2
VG Access        read/write
VG Status        resizable
MAX LV           0
Cur LV           1
Open LV           1
Max PV           0
Cur PV           2
Act PV           2
VG Size          1012.00 MiB  ②
PE Size          4.00 MiB
Total PE         253          ③
Alloc PE / Size  75 / 300.00 MiB
Free PE / Size   178 / 712.00 MiB ④
VG UUID          jK5M1M-YvIk-kxU2-bxmS-dNjQ-Bs3L-DRljNc
```

- ❶ **VG Name**은 볼륨 그룹의 이름을 표시합니다.
- ❷ **VG Size**는 LV 할당에 사용 가능한 스토리지 풀의 총 크기입니다.
- ❸ **Total PE**는 PE 장치의 총 크기를 표시합니다.
- ❹ **Free PE / Size**는 VG에서 LV를 생성하거나 확장하는 데 사용할 수 있는 공간을 표시합니다.

논리 볼륨 정보 표시

lvdisplay 명령은 논리 볼륨에 대한 정보를 표시합니다. 이 명령을 인수 없이 사용하면 모든 PV의 정보가 나열됩니다. LV 이름을 인수로 제공하면 해당 LV와 관련된 정보가 표시됩니다.

```
[root@host ~]# lvdisplay /dev/vg01/lv01
--- Logical volume ---
LV Path          /dev/vg01/lv01          ❶
LV Name          lv01
VG Name          vg01          ❷
LV UUID          FVmNel-u25R-dt3p-C5L6-VP2w-QRNP-scqrbq
LV Write Access  read/write
LV Creation host, time servera.lab.example.com, 2022-04-07 10:45:34 -0400
LV Status        available
# open           1
LV Size          300.00 MiB          ❸
Current LE       75          ❹
Segments         1
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

- ❶ **LV Path**는 LV의 장치 이름을 표시합니다.
- ❷ **VG Name**은 이 LV를 생성하는 데 사용된 VG를 표시합니다.
- ❸ **LV Size**는 LV의 총 크기를 표시합니다. LV에 사용 가능한 공간과 사용된 공간을 확인하려면 파일 시스템 툴을 사용합니다.
- ❹ **Current LE**는 이 LV에서 사용한 논리 확장 영역 수를 표시합니다.

LVM 스토리지 확장 및 축소

논리 볼륨을 생성한 뒤 볼륨을 확장하여 파일 시스템을 확장할 수 있습니다. LV의 스토리지 용량을 늘리기 위해 PV 또는 VG를 확장해야 할 수 있습니다.

볼륨 그룹(VG) 크기 확장

VG를 확장하면서 더 많은 디스크 공간을 추가해야 할 수 있습니다. VG에 물리 볼륨을 추가하여 사용 가능한 크기를 확장할 수 있습니다.

물리 장치를 준비하고 물리 볼륨이 없는 경우 이를 생성합니다.

```
[root@host ~]# parted /dev/vdb mkpart primary 1072MiB 1648MiB
...output omitted...
[root@host ~]# parted /dev/vdb set 3 lvm on
...output omitted...
[root@host ~]# udevadm settle
[root@host ~]# pvcreate /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
```

vgextend 명령은 VG에 새 PV를 추가합니다. VG 및 PV 이름을 **vgextend** 명령에 인수로 제공합니다.

```
[root@host ~]# vgextend vg01 /dev/vdb3
Volume group "vg01" successfully extended
```

이 명령은 **vg01** VG를 **/dev/vdb3** PV 크기만큼 확장합니다.

논리 볼륨 크기 확장

논리 볼륨을 사용할 때의 이점은 다운타임 없이 크기를 늘릴 수 있다는 것입니다. LV에서 VG로 사용 가능한 물리적 확장 영역을 추가하여 LV 용량을 확장하고 파일 시스템을 확장합니다. **vgdisplay** 명령을 사용하여 볼륨 그룹에 LV 확장 영역을 위한 여유 공간이 충분히 있는지 확인합니다.

lvextend 명령을 사용하여 LV를 확장합니다.

```
[root@host ~]# lvextend -L +500M /dev/vg01/lv01
Size of logical volume vg01/lv01 changed from 300.00 MiB (75 extents) to 800.00
MiB (200 extents).
Logical volume vg01/lv01 successfully resized.
```

이 명령은 **lv01** 논리 볼륨의 크기를 500MiB 늘립니다. 크기 앞에 있는 더하기 표시(+)는 이 값을 기존 크기에 추가한다는 의미이며 이 표시가 없는 경우 해당 값은 최종 LV 크기를 정의합니다.

lvextend 명령 -L 옵션에는 인수로 PE 수가 있어야 합니다. **lvextend** 명령 -L 옵션에는 바이트, 메비바이트, 기비바이트 등의 단위로 된 크기가 사용됩니다.

XFS 파일 시스템을 논리 볼륨 크기로 확장

xfs_growfs 명령은 파일 시스템을 확장하여 확장된 LV를 차지하는 데 도움이 됩니다. **xfs_growfs** 명령을 사용하기 전에 타겟 파일 시스템을 마운트해야 합니다. 크기를 조정하는 동안 파일 시스템을 계속 사용할 수 있습니다.

```
[root@host ~]# xfs_growfs /mnt/data/
...output omitted...
data blocks changed from 76800 to 204800
```



중요

lvextend 명령을 실행한 후에는 항상 **xfs_growfs** 명령을 실행하십시오. **lvextend** 명령 -r 옵션을 사용하여 두 단계를 연속적으로 실행합니다. LV를 확장한 후 **fsadm** 명령을 사용하여 파일 시스템의 크기를 조정합니다. 이 옵션은 다양한 기타 파일 시스템을 지원합니다.

EXT4 파일 시스템을 논리 볼륨 크기로 확장

ext4 파일 시스템을 사용하여 LV를 확장하는 단계는 파일 시스템 크기를 조정하는 단계를 제외하고 **XFS** 파일 시스템을 사용하는 LV와 동일합니다.

resize2fs 명령은 파일 시스템을 확장하여 확장된 새 LV를 차지합니다. 크기를 조정하는 동안 파일 시스템을 계속 사용할 수 있습니다.

```
[root@host ~]# resize2fs /dev/vg01/lv01
resize2fs 1.46.5 (30-Dec-2021)
Resizing the filesystem on /dev/vg01/lv01 to 256000 (4k) blocks.
The filesystem on /dev/vg01/lv01 is now 256000 (4k) blocks long.
```

xfs_growfs 와 **resize2fs** 의 주요 차이점은 파일 시스템을 식별하는데 전달되는 인수입니다. **xfs_growfs** 명령은 마운트 지점을 인수로 사용하고, **resize2fs** 명령은 LV 이름을 인수로 사용합니다. **xfs_growfs** 명령은 온라인 크기 조정만 지원하지만 **resize2fs** 명령은 온라인 및 오프라인 크기 조정을 모두 지원합니다. **ext4** 파일 시스템은 크기를 늘리거나 줄일 수 있지만 **XFS** 파일 시스템은 늘릴 수만 있습니다.

스왑 공간 논리 볼륨 확장

스왑 공간으로 사용되는 LV를 확장 할 수 있습니다. 그러나 해당 프로세스는 **ext4** 또는 **XFS** 파일 시스템을 확장하는 것과는 다릅니다. 스왑 공간으로 사용되는 논리 볼륨은 오프라인 상태에서 확장해야 합니다.

swapoff 명령을 사용하여 LV에서 새 스왑 공간을 비활성화하십시오.

```
[root@host ~]# swapoff -v /dev/vg01/swap
swapoff /dev/vg01/swap
```

lvextend 명령을 사용하여 LV를 확장합니다.

```
[root@host ~]# lvextend -L +300M /dev/vg01/swap
  Size of logical volume vg01/swap changed from 500.00 MiB (125 extents) to 800.00
  MiB (200 extents).
  Logical volume vg01/swap successfully resized.
```

mkswap 명령을 사용하여 LV를 스왑 공간으로 포맷합니다.

```
[root@host ~]# mkswap /dev/vg01/swap
mkswap: /dev/vg01/swap: warning: wiping old swap signature.
Setting up swapspace version 1, size = 800 MiB (838856704 bytes)
no label, UUID=25b4d602-6180-4b1c-974e-7f40634ad660
```

swapon 명령을 사용하여 LV에서 새 스왑 공간을 활성화합니다.

```
[root@host ~]# swapon /dev/vg01/swap
```

볼륨 그룹 스토리지 줄이기

VG를 줄이려면 VG에서 사용되지 않은 PV를 제거해야 합니다. **pvmove** 명령은 특정 PV의 확장 영역에 있는 데이터를 동일한 VG에서 사용 가능한 확장 영역이 충분한 다른 PV의 확장 영역으로 이동합니다. 축소하는 동

안 동일한 VG의 LV를 계속 사용할 수 있습니다. 변경 후 VG의 메타데이터를 자동으로 백업하려면 **pvmove** 명령 -A 옵션을 사용합니다. 이 옵션은 **vgcfgbackup** 명령을 사용하여 메타데이터를 자동으로 백업합니다.

```
[root@host ~]# pvmove /dev/vdb3
```



경고

pvmove 명령을 사용하기 전에 VG의 모든 LV에 저장된 데이터를 백업하십시오. 작업 중에 예기치 않은 정전이 발생하면 VG가 일관되지 않은 상태가 되어 LV의 데이터가 손실될 수 있습니다.

vgreduce 명령을 사용하여 VG에서 PV를 제거합니다.

```
[root@host ~]# vgreduce vg01 /dev/vdb3
Removed "/dev/vdb3" from volume group "vg01"
```



중요

GFS2 및 XFS 파일 시스템은 축소를 지원하지 않으므로 LV의 크기를 줄일 수 없습니다.

LVM 스토리지 제거

더 이상 필요하지 않은 LVM 구성 요소를 제거하려면 **lvremove**, **vgremove**, **pvremove** 명령을 사용합니다.

파일 시스템 준비

유지해야 할 모든 데이터를 다른 파일 시스템으로 이동합니다. **umount** 명령을 사용하여 파일 시스템을 마운트 해제한 다음 이 파일 시스템과 관련된 **/etc/fstab** 항목을 모두 제거합니다.

```
[root@host ~]# umount /mnt/data
```



경고

논리 볼륨을 제거하면 논리 볼륨에 저장된 모든 데이터가 삭제됩니다. 논리 볼륨을 제거하기 전에 데이터를 백업 또는 이동합니다.

논리 볼륨 제거

더 이상 필요하지 않은 논리 볼륨을 제거하려면 **lvremove DEVICE-NAME** 명령을 사용합니다. 이 명령을 실행하기 전에 LV 파일 시스템을 마운트 해제해야 합니다. 이 명령은 LV를 제거하기 전에 확인 메시지를 표시합니다.

```
[root@host ~]# lvremove /dev/vg01/lv01
Do you really want to remove active logical volume vg01/lv01? [y/n]: y
Logical volume "lv01" successfully removed.
```

LV의 물리 확장 영역은 해제되어 볼륨 그룹의 기존 LV 또는 새 LV에 할당할 수 있습니다.

볼륨 그룹 제거

더 이상 필요하지 않은 볼륨 그룹을 제거하려면 **vgremove VG-NAME** 명령을 사용합니다.

```
[root@host ~]# vgremove vg01
Volume group "vg01" successfully removed
```

VG의 물리 볼륨을 사용할 수 있고 시스템의 기존 또는 새 VG에 할당할 수 있게 됩니다.

물리 볼륨 제거

더 이상 필요하지 않은 물리 볼륨을 제거하려면 **pvremove** 명령을 사용합니다. 한 번에 둘 이상의 장치를 제거하려면 공백으로 구분된 PV 장치 목록을 사용합니다. 이 명령을 실행하면 파티션(또는 디스크)에서 PV 메타데이터가 삭제됩니다. 이제 파티션을 재할당 또는 재포맷할 수 있습니다.

```
[root@host ~]# pvremove /dev/vdb1 /dev/vdb2
Labels on physical volume "/dev/vdb1" successfully wiped.
Labels on physical volume "/dev/vdb2" successfully wiped.
```



참조

fdisk(8), gdisk(8), parted(8), partprobe(8), lvm(8), pvcreate(8), vgcreate(8), lvcreate(8), mkfs(8), pvdisk(8), vgdisplay(8), lvdisplay(8), vgextend(8), lvextend(8), xfs_growfs(8), resize2fs(8), swapoff(8), mkswap(8), swapon(8), pmove(8), vgcfgbackup(8), vgreduce(8), lvremove(8), vgremove(8), pvremove(8) 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_logical_volumes/index
에서 Configuring and Managing Logical Volumes 를 참조하십시오.

▶ 연습 가이드

논리 볼륨 생성 및 확장

이 랩에서는 물리 볼륨, 볼륨 그룹, 논리 볼륨, XFS 파일 시스템을 생성하고 확장합니다. 또한 논리 볼륨 파일 시스템을 영구적으로 마운트합니다.

결과

- LVM 도구를 사용하여 물리 볼륨, 볼륨 그룹, 논리 볼륨을 만듭니다.
- 논리 볼륨에 파일 시스템을 만들고 영구적으로 마운트합니다.
- 볼륨 그룹을 확장하여 추가 물리 볼륨을 포함합니다.
- 파일 시스템이 마운트되어 있고 사용 중인 상태로 논리 볼륨의 크기를 조정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start lvm-manage
```

지침

▶ 1. servera 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

▶ 2. **/dev/vdb** 스토리지 장치에 물리 장치 파티션을 생성합니다.

- 2.1. 각각 256MiB 파티션 두 개를 생성하고 Linux LVM 유형으로 설정합니다. 이러한 파티션의 이름으로 **first** 및 **second**를 사용합니다.

```
[root@servera ~]# parted /dev/vdb mklabel gpt
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb mkpart first 1MiB 258MiB
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb set 1 lvm on
Information: You may need to update /etc/fstab.
```

```
[root@servera ~]# parted /dev/vdb mkpart second 258MiB 514MiB
Information: You may need to update /etc/fstab.

[root@servera ~]# parted /dev/vdb set 2 lvm on
Information: You may need to update /etc/fstab.
```

2.2. 커널에 새 파티션을 등록합니다.

```
[root@servera ~]# udevadm settle
```

2.3. `/dev/vdb` 스토리지 장치의 파티션을 나열합니다. `Number` 열에서 1 및 2 값은 `/dev/vdb1` 및 `/dev/vdb2` 장치 파티션에 해당합니다. `Flags` 열은 파티션 유형을 나타냅니다.

```
[root@servera ~]# parted /dev/vdb print
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 5369MB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size   File system  Name    Flags
 1      1049kB  271MB  269MB          first    lvm
 2      271MB   539MB  268MB          second   lvm
```

▶ 3. 두 개의 새 파티션에 물리 볼륨으로 레이블을 지정합니다.

```
[root@servera ~]# pvcreate /dev/vdb1 /dev/vdb2
Physical volume "/dev/vdb1" successfully created.
Physical volume "/dev/vdb2" successfully created.
Creating devices file /etc/lvm/devices/system.devices
```

▶ 4. 새 PV 두 개를 사용하여 `servera_group` 볼륨 그룹을 생성합니다.

```
[root@servera ~]# vgcreate servera_group /dev/vdb1 /dev/vdb2
Volume group "servera_group" successfully created
```

▶ 5. 크기가 400MiB인 `servera_volume` 논리 볼륨을 생성합니다. 이 명령은 파일 시스템 없이 `/dev/servera_group/servera_volume` LV를 생성합니다.

```
[root@servera ~]# lvcreate -n servera_volume -L 400M servera_group
Logical volume "servera_volume" created.
```

▶ 6. 새로 생성한 LV를 포맷하고 영구적으로 마운트합니다.

6.1. `servera_volume` LV를 XFS 파일 시스템으로 포맷합니다.

```
[root@servera ~]# mkfs -t xfs /dev/servera_group/servera_volume
...output omitted...
```

6.2. `/data` 디렉터리를 마운트 지점으로 생성합니다.

```
[root@servera ~]# mkdir /data
```

- 6.3. 새로 생성된 파일 시스템을 영구적으로 마운트하려면 **/etc/fstab** 파일에 다음 내용을 추가합니다.

```
/dev/servera_group/servera_volume /data xfs defaults 0 0
```

- 6.4. **servera_volume** LV를 마운트합니다.

```
[root@servera ~]# mount /data
```

▶ 7. 마운트한 파일 시스템에 액세스할 수 있는지 확인하고 LVM의 상태 정보를 표시합니다.

- 7.1. 파일을 **/data** 디렉터리에 복사할 수 있는지 확인합니다.

```
[root@servera ~]# cp -a /etc/*.* /data
[root@servera ~]# ls /data | wc -l
32
```

- 7.2. PV 상태 정보를 확인합니다. 출력에 PV에서 **servera_group** VG를 사용하는 것으로 표시됩니다. PV의 크기는 256MiB이고 물리 확장 영역 크기는 4MiB입니다.

VG에 63개의 PE가 있으며 27개의 PE를 할당할 수 있고, 현재 36개의 PE가 LV에 할당되어 있습니다. 볼륨 크기를 MiB 단위로 할당하려면 다음 계산을 사용합니다.

- 총 252MiB(PE 63개 x 4MiB)
- 108MiB 사용 가능(PE 27개 x 4MiB)
- 144MiB 할당(PE 36개 x 4MiB)

```
[root@servera ~]# pvdisplay /dev/vdb2
--- Physical volume ---
PV Name           /dev/vdb2
VG Name           servera_group
PV Size          256.00 MiB / not usable 4.00 MiB
Allocatable       yes
PE Size          4.00 MiB
Total PE         63
Free PE          27
Allocated PE     36
PV UUID          FKKFYJ-wJiR-1jt2-sfy3-yjPy-TylN-LG92jj
```

- 7.3. **servera_group** VG의 VG 상태 정보를 확인합니다. 출력에 VG 크기 508 MiB 및 PE 크기 4 MiB가 표시됩니다. VG에서 사용 가능한 크기는 108 MiB입니다.

```
[root@servera ~]# vgdisplay servera_group
--- Volume group ---
VG Name           servera_group
System ID
Format           lvm2
Metadata Areas   2
Metadata Sequence No  2
```

```

VG Access          read/write
VG Status          resizable
MAX LV             0
Cur LV             1
Open LV            1
Max PV             0
Cur PV             2
Act PV             2
VG Size            508.00 MiB
PE Size             4.00 MiB
Total PE           127
Alloc PE / Size   100 / 400.00 MiB
Free PE / Size    27 / 108.00 MiB
VG UUID            g0ahyT-90J5-iGic-nnb5-G6T9-tLdk-dX8c9M

```

- 7.4. `servera_volume` LV의 상태 정보를 확인합니다. 출력에 LV를 생성하기 위한 VG 이름이 표시됩니다. 또한 LV 크기 400 MiB 및 LE 크기 100이 표시됩니다.

```

[root@servera ~]# lvdisplay /dev/servera_group/servera_volume
--- Logical volume ---
LV Path              /dev/servera_group/servera_volume
LV Name              servera_volume
VG Name              servera_group
LV UUID              93MfUt-esgT-B5HM-r1p5-DVZH-n5cn-J5e2tw
LV Write Access      read/write
LV Creation host, time servera.lab.example.com, 2022-04-11 03:25:12 -0400
LV Status            available
# open               1
LV Size              400.00 MiB
Current LE           100
Segments             2
Allocation           inherit
Read ahead sectors  auto
- currently set to  8192
Block device         253:0

```

- 7.5. 사용 가능한 디스크 공간을 사람이 읽을 수 있는 단위로 봅니다. 출력에 총 크기 395MiB 및 사용 가능한 크기 372MiB가 표시됩니다.

```

[root@servera ~]# df -h /data
Filesystem           Size  Used Avail Use% Mounted on
/dev/mapper/servera_group-servera_volume 395M   24M  372M   6% /data

```

▶ 8. /dev/vdb 스토리지 장치에 물리 리소스를 생성합니다.

- 8.1. 512MiB의 추가 파티션을 생성하고 Linux LVM 유형으로 설정합니다. 파티션 이름으로 `third`를 사용합니다.

```

[root@servera ~]# parted /dev/vdb mkpart third 514MiB 1026MiB
[root@servera ~]# parted /dev/vdb set 3 lvm on

```

- 8.2. 커널에 새 파티션을 등록합니다.

```
[root@servera ~]# udevadm settle
```

8.3. 새 파티션을 PV로 추가합니다.

```
[root@servera ~]# pvcreate /dev/vdb3
Physical volume "/dev/vdb3" successfully created.
```

- ▶ 9. 새로 생성된 디스크 공간을 사용하여 `servera_volume`의 파일 시스템 크기를 총 700MiB로 확장합니다.

9.1. 새 `/dev/vdb3` PV를 사용하여 `servera_group` VG를 확장합니다.

```
[root@servera ~]# vgextend servera_group /dev/vdb3
Volume group "servera_group" successfully extended
```

9.2. 기존 `servera_volume` LV를 700MiB로 확장합니다.

```
[root@servera ~]# lvextend -L 700M /dev/servera_group/servera_volume
Size of logical volume servera_group/servera_volume changed from 400.00 MiB (100
extents) to 700.00 MiB (175 extents).
Logical volume servera_group/servera_volume successfully resized.
```

9.3. LV의 사용 가능한 공간을 사용하여 XFS 파일 시스템을 확장합니다.

```
[root@servera ~]# xfs_growfs /data
...output omitted...
data blocks changed from 102400 to 179200
```

- ▶ 10. LV 크기가 확장되었으며 콘텐츠가 여전히 해당 볼륨에 있는지 확인합니다.

10.1. `lvdisplay` 명령을 사용하여 확장된 LV의 크기를 확인합니다.

```
[root@servera ~]# lvdisplay /dev/servera_group/servera_volume
--- Logical volume ---
LV Path          /dev/servera_group/servera_volume
LV Name          servera_volume
VG Name          servera_group
LV UUID          mLQhsD-hyL0-KC2B-2nug-o2Nc-0znS-Q428fK
LV Write Access  read/write
LV Creation host, time servera.lab.example.com, 2022-04-12 06:04:12 -0400
LV Status        available
# open           1
LV Size          700.00 MiB
Current LE       175
Segments         3
Allocation       inherit
Read ahead sectors auto
- currently set to 8192
Block device     253:0
```

10.2. 새 파일 시스템 크기를 확인합니다. 이전에 복사한 파일이 여전히 있는지 확인합니다.

```
[root@servera ~]# df -h /data
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/servera_group-servera_volume  695M   26M  670M   4% /data
[root@servera ~]# ls /data | wc -l
32
```

▶ 11. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish lvm-manage
```

이것으로 섹션을 완료합니다.

▶ 랩

스토리지 스택 관리

이 랩에서는 기존 논리 볼륨의 크기를 조정하고 필요에 따라 LVM 리소스를 추가한 다음 XFS 파일 시스템이 영구적으로 마운트된 새 논리 볼륨을 추가합니다.

결과

- serverb_01_lv** 논리 볼륨의 크기를 768MiB로 조정합니다.
- XFS 파일 시스템을 사용하여 128MiB의 **serverb_02_lv** 논리 볼륨을 생성합니다.
- 해당 볼륨을 **/storage/data2** 디렉터리에 영구적으로 마운트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start lvm-review
```

지침

serverb 시스템에서 **/storage/data1** 디렉터리에 마운트된 **serverb_01_lv** 논리 볼륨의 디스크 공간이 부족하여 768MiB로 확장해야 합니다. **serverb_01_lv** LV가 **/storage/data1** 디렉터리에 영구적으로 마운트되어 있는지 확인해야 합니다.

serverb_01_lv LV는 **serverb_01_vg** 볼륨 그룹에 있습니다. 기존 논리 볼륨을 확장할 공간이 충분하지 않습니다. **/dev/vdb** 디스크에 512MiB의 파티션이 있습니다. **/dev/vdb** 디스크에 512MiB 파티션을 생성합니다.



중요

MiB(2^{20} 바이트) 단위의 파티션 크기 사양에 주의하십시오. 파티션을 MB(10^6 바이트) 단위로 생성하는 경우 1MiB = 1.048576MB이므로 평가 기준을 충족하지 않습니다.

parted /dev/vdb print 명령을 사용할 때의 기본 단위는 MB이지만 **/dev/vdb** 장치 파티션의 크기를 MiB 단위로 확인할 수 있습니다. 파티션 크기를 MiB 단위로 출력하려면 **parted /dev/vdb unit MiB print** 명령을 사용하십시오.

128MiB로 **serverb_02_lv** LV를 생성합니다. 새로 생성한 볼륨에 XFS 파일 시스템을 생성합니다. 새로 생성한 논리 볼륨을 **/storage/data2** 디렉터리에 마운트합니다.

- /dev/vdb** 디스크에 512MiB 파티션을 생성합니다. 이 파티션을 물리 볼륨으로 초기화하고 **serverb_01_vg** 볼륨 그룹을 확장하여 이 파티션을 사용합니다.
- serverb_01_lv** 논리 볼륨의 크기를 768MiB로 확장합니다.

3. 기존 볼륨 그룹에서 128MiB의 **serverb_02_lv** 논리 볼륨을 생성합니다. XFS 파일 시스템을 추가하고 **/storage/data2** 디렉터리에 영구적으로 마운트합니다.
4. 새로 생성된 LV가 원하는 크기로 마운트되었는지 확인합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade lvm-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish lvm-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

스토리지 스택 관리

이 랩에서는 기존 논리 볼륨의 크기를 조정하고 필요에 따라 LVM 리소스를 추가한 다음 XFS 파일 시스템이 영구적으로 마운트된 새 논리 볼륨을 추가합니다.

결과

- serverb_01_lv** 논리 볼륨의 크기를 768MiB로 조정합니다.
- XFS 파일 시스템을 사용하여 128MiB의 **serverb_02_lv** 논리 볼륨을 생성합니다.
- 해당 볼륨을 **/storage/data2** 디렉터리에 영구적으로 마운트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start lvm-review
```

지침

serverb 시스템에서 **/storage/data1** 디렉터리에 마운트된 **serverb_01_lv** 논리 볼륨의 디스크 공간이 부족하여 768MiB로 확장해야 합니다. **serverb_01_lv** LV가 **/storage/data1** 디렉터리에 영구적으로 마운트되어 있는지 확인해야 합니다.

serverb_01_lv LV는 **serverb_01_vg** 볼륨 그룹에 있습니다. 기존 논리 볼륨을 확장할 공간이 충분하지 않습니다. **/dev/vdb** 디스크에 512MiB의 파티션이 있습니다. **/dev/vdb** 디스크에 512MiB 파티션을 생성합니다.



중요

MiB(2^{20} 바이트) 단위의 파티션 크기 사양에 주의하십시오. 파티션을 MB(10^6 바이트) 단위로 생성하는 경우 1MiB = 1.048576MB이므로 평가 기준을 충족하지 않습니다.

parted /dev/vdb print 명령을 사용할 때의 기본 단위는 MB이지만 **/dev/vdb** 장치 파티션의 크기를 MiB 단위로 확인할 수 있습니다. 파티션 크기를 MiB 단위로 출력하려면 **parted /dev/vdb unit MiB print** 명령을 사용하십시오.

128MiB로 **serverb_02_lv** LV를 생성합니다. 새로 생성한 볼륨에 XFS 파일 시스템을 생성합니다. 새로 생성한 논리 볼륨을 **/storage/data2** 디렉터리에 마운트합니다.

- /dev/vdb** 디스크에 512MiB 파티션을 생성합니다. 이 파티션을 물리 볼륨으로 초기화하고 **serverb_01_vg** 볼륨 그룹을 확장하여 이 파티션을 사용합니다.

1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

1.2. 파티션 크기를 MiB 단위로 인쇄하여 첫 번째 파티션이 끝나는 위치를 확인합니다.

```
[root@serverb ~]# parted /dev/vdb unit MiB print
...output omitted...

Number  Start   End     Size   File system  Name      Flags
 1       1.00MiB 513MiB 512MiB          primary
```

1.3. 512MiB 파티션을 생성하고 lvm 파티션 유형을 설정합니다.

```
[root@serverb ~]# parted /dev/vdb mkpart primary 514MiB 1026MiB
...output omitted...
[root@serverb ~]# parted /dev/vdb set 2 lvm on
```

1.4. 커널에 새 파티션을 등록합니다.

```
[root@serverb ~]# udevadm settle
```

1.5. 파티션을 PV로 초기화합니다.

```
[root@serverb ~]# pvcreate /dev/vdb2
Physical volume "/dev/vdb2" successfully created.
```

1.6. 새 /dev/vdb2 PV를 사용하여 serverb_01_vg VG를 확장합니다.

```
[root@serverb ~]# vgextend serverb_01_vg /dev/vdb2
Volume group "serverb_01_vg" successfully extended
```

2. serverb_01_lv 논리 볼륨의 크기를 768MiB로 확장합니다.

2.1. serverb_01_lv LV를 768MiB로 확장합니다.

또는 lvcreate 명령 -L +512M 옵션을 사용하여 LV의 크기를 조정할 수 있습니다.

```
[root@serverb ~]# lvextend -L 768M /dev/serverb_01_vg/serverb_01_lv
Size of logical volume serverb_01_vg/serverb_01_lv changed from 256.00 MiB (64
extents) to 768.00 MiB (192 extents).
Logical volume serverb_01_vg/serverb_01_lv successfully resized.
```

2.2. LV의 나머지 공간을 사용하는 XFS 파일 시스템을 확장합니다.

```
[root@serverb ~]# xfs_growfs /storage/data1
meta-data=/dev/mapper/serverb_01_vg-serverb_01_lv isize=512    agcount=4,
agsize=16384 blks
...output omitted...
data blocks changed from 65536 to 196608
```

**참고**

`xfs_growfs` 명령은 파일 시스템을 확장하는 추가 단계를 적용합니다. 대안은 `lvextend` 명령 `-r` 옵션을 사용하는 것입니다.

3. 기존 볼륨 그룹에서 128MiB의 `serverb_02_lv` 논리 볼륨을 생성합니다. XFS 파일 시스템을 추가하고 `/storage/data2` 디렉터리에 영구적으로 마운트합니다.

- 3.1. `serverb_01_vg` VG에서 128MiB의 `serverb_02_lv` LV를 생성합니다.

```
[root@serverb ~]# lvcreate -n serverb_02_lv -L 128M serverb_01_vg
Logical volume "serverb_02_lv" created.
```

- 3.2. `serverb_02_lv` LV에 xfs 파일 시스템을 생성합니다.

```
[root@serverb ~]# mkfs -t xfs /dev/serverb_01_vg/serverb_02_lv
...output omitted...
```

- 3.3. `/storage/data2` 디렉터리를 마운트 지점으로 생성합니다.

```
[root@serverb ~]# mkdir /storage/data2
```

- 3.4. 다음 행을 `/etc/fstab` 파일의 끝에 추가합니다.

```
/dev/serverb_01_vg/serverb_02_lv /storage/data2 xfs defaults 0 0
```

- 3.5. 새 `/etc/fstab` 구성 파일로 `systemd` 데몬을 업데이트합니다.

```
[root@serverb ~]# systemctl daemon-reload
```

- 3.6. `serverb_02_lv` LV를 마운트합니다.

```
[root@serverb ~]# mount /storage/data2
```

4. 새로 생성된 LV가 원하는 크기로 마운트되었는지 확인합니다.

- 4.1. `df` 명령을 사용하여 `serverb_01_lv` LV 크기를 확인합니다.

```
[root@serverb ~]# df -h /storage/data1
Filesystem           Size   Used  Avail Use% Mounted on
/dev/mapper/serverb_01_vg-serverb_01_lv  763M   19M  744M   3% /storage/data1
```

4.2. `serverb_02_lv` LV 크기를 확인합니다.

```
[root@serverb ~]# df -h /storage/data2
Filesystem                      Size  Used Avail Use% Mounted on
/dev/mapper/serverb_01_vg-serverb_02_lv  123M  7.6M  116M   7% /storage/data2
```

4.3. `serverb_01_lv` LV 세부 정보를 확인합니다.

```
[root@serverb ~]# lvdisplay /dev/serverb_01_vg/serverb_01_lv
--- Logical volume ---
LV Path                  /dev/serverb_01_vg/serverb_01_lv
LV Name                 serverb_01_lv
VG Name                 serverb_01_vg
LV UUID                 1pY3DZ-fs1F-mptC-fL32-e8tG-PFBT-bs7LSJ
LV Write Access          read/write
LV Creation host, time  serverb.lab.example.com, 2022-05-05 14:40:51 -0400
LV Status                available
# open                  1
LV Size                  768.00 MiB
Current LE               192
Segments                2
Allocation              inherit
Read ahead sectors      auto
- currently set to     8192
Block device             253:0
```

4.4. `serverb_02_lv` LV 세부 정보를 확인합니다.

```
[root@serverb ~]# lvdisplay /dev/serverb_01_vg/serverb_02_lv
--- Logical volume ---
LV Path                  /dev/serverb_01_vg/serverb_02_lv
LV Name                 serverb_02_lv
VG Name                 serverb_01_vg
LV UUID                 0aJIB6-Ti2b-jLCK-imB6-rkLx-mUoX-acjkz9
LV Write Access          read/write
LV Creation host, time  serverb.lab.example.com, 2022-05-05 14:45:46 -0400
LV Status                available
# open                  1
LV Size                  128.00 MiB
Current LE               32
Segments                1
Allocation              inherit
Read ahead sectors      auto
- currently set to     8192
Block device             253:1
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade lvm-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish lvm-review
```

이것으로 섹션을 완료합니다.

요약

- LVM을 사용하면 여러 스토리지 장치에 공간을 할당하여 유연한 스토리지를 만들 수 있습니다.
- 물리 볼륨, 볼륨 그룹, 논리 볼륨은 **pvccreate**, **vgreduce**, **lvextend** 명령으로 관리할 수 있습니다.
- 논리 볼륨은 파일 시스템 또는 스왑 공간으로 포맷하고 영속적으로 마운트할 수 있습니다.
- 볼륨 그룹에 스토리지를 추가할 수 있으며 논리 볼륨은 동적으로 확장할 수 있습니다.

11장

제어 서비스 및 부팅 프로세스

목적

systemd 서비스를 사용하여 네트워크 서비스, 시스템 데몬 및 부팅 프로세스를 제어 및 모니터링합니다.

목표

- **systemd** 서비스 및 소켓 유닛에서 시작된 시스템 데몬과 네트워크 서비스를 표시합니다.
- **systemctl**을 사용하여 시스템 데몬 및 네트워크 서비스를 제어합니다.
- Red Hat Enterprise Linux 부팅 프로세스를 설명하고, 부팅 시 기본 타겟을 설정하고, 시스템을 기본값이 아닌 타겟으로 부팅합니다.
- 현재 루트 암호가 분실된 경우 시스템에 로그인하여 루트 암호를 변경합니다.
- 부팅 프로세스를 중단하는 파일 시스템 구성 또는 손상 문제를 수동으로 복구합니다.

섹션

- 자동으로 시작된 시스템 프로세스 식별(안내에 따른 연습)
- 시스템 서비스 제어(안내에 따른 연습)
- 부팅 타겟 선택(안내에 따른 연습)
- 루트 암호 재설정(안내에 따른 연습)
- 부팅 시 파일 시스템 문제 복구(안내에 따른 연습)

랩

- 부팅 프로세스 제어

자동으로 시작된 시스템 프로세스 식별

목표

systemd 서비스 및 소켓 유닛에서 시작된 시스템 데몬과 네트워크 서비스를 표시합니다.

systemd 데몬 소개

systemd 데몬은 일반적인 서비스 시작 및 서비스 관리를 포함하여 Linux 시작 프로세스를 관리합니다.

systemd 데몬은 부팅할 때 및 실행 중인 시스템에서 시스템 리소스, 서버 데몬, 기타 프로세스를 활성화합니다.

데몬은 백그라운드에서 대기하거나 실행되어 다양한 작업을 수행하는 프로세스입니다. 일반적으로 데몬은 부팅 시 자동으로 시작되며, 종료되거나 수동으로 중지할 때까지 계속 실행됩니다. 규칙에 따라 데몬 이름은 **d**로 끝납니다.

systemd에서 서비스는 하나 이상의 데몬을 가리키는 경우가 많습니다. 그러나 서비스를 시작하거나 중지할 때 실행 중인 데몬 프로세스가 남는 대신 시스템 상태가 한 번 변경될 수도 있습니다(**oneshot**이라고 함).

Red Hat Enterprise Linux에서 시작되는 첫 번째 프로세스(PID 1)는 **systemd** 데몬으로, 다음 기능을 제공합니다.

- 시스템 부팅 속도를 높이는 병렬화 기능(동시에 여러 서비스 시작)
- 별도의 서비스를 요구하지 않고 필요할 때 데몬 시작
- 긴 타임아웃을 방지할 수 있는 자동 서비스 종속성 관리. 예를 들어 네트워크 종속 서비스는 네트워크를 사용할 수 있을 때까지 시작되지 않습니다.
- Linux 제어 그룹을 사용하여 관련 프로세스를 한꺼번에 추적하는 방법

서비스 유닛 설명

systemd 유닛은 시스템에서 관리 방법을 알고 있는 오브젝트를 정의하는데 사용되는 추상적인 개념입니다.

유닛은 시스템 서비스, 수신 소켓, **systemd** init 시스템과 관련된 기타 오브젝트에 대한 정보를 캡슐화하는 유닛 파일이라는 구성 파일로 표시됩니다.

유닛에는 이름과 유닛 유형이 있습니다. 이름은 유닛에 고유한 ID를 제공합니다. 유닛 유형을 사용하면 유닛을 다른 유사한 유닛 유형과 함께 그룹화할 수 있습니다.

systemd 데몬은 **유닛**을 사용하여 다양한 유형의 오브젝트를 관리합니다.

- 서비스 유닛의 확장자는 **.service**이며 시스템 서비스를 나타냅니다. 서비스 유닛을 사용하여 웹 서버와 같이 자주 액세스하는 데몬을 시작할 수 있습니다.
- 소켓 유닛의 확장자는 **.socket**이며 **systemd**에서 모니터링해야 하는 프로세스 내 통신(IPC) 소켓을 나타냅니다. 클라이언트가 소켓에 연결하면 **systemd**가 데몬을 시작하고 데몬에 연결을 전달합니다. 소켓 유닛을 사용하여 부팅 시 서비스 시작을 연기하고 자주 사용되지 않는 서비스를 요청할 시 시작할 수 있습니다.
- 경로 유닛은 확장자가 **.path**이며 특정 파일 시스템 변경이 발생할 때까지 서비스 활성화를 연기합니다. 출력 시스템과 같이 스펄 디렉터리를 사용하는 서비스에 경로 유닛을 사용할 수 있습니다.

유닛을 관리하려면 `systemctl` 명령을 사용합니다. 예를 들어 `systemctl -t help` 명령을 사용하여 사용 가능한 유닛 유형을 표시합니다. `systemctl` 명령은 약식 유닛 이름, 프로세스 트리 항목, 유닛 설명을 사용할 수 있습니다.

서비스 유닛 표시

`systemctl` 명령을 사용하여 시스템의 현재 상태를 살펴봅니다. 예를 들어 다음 명령은 현재 로드된 모든 서비스 유닛을 표시하고 페이지를 매깁니다.

```
[root@host ~]# systemctl list-units --type=service
UNIT                  LOAD    ACTIVE   SUB      DESCRIPTION
atd.service           loaded  active   running  Job spooling tools
auditd.service        loaded  active   running  Security Auditing Service
chronyd.service       loaded  active   running  NTP client/server
crond.service         loaded  active   running  Command Scheduler
dbus.service          loaded  active   running  D-Bus System Message Bus
...output omitted...
```

이 예제에서 `--type=service` 옵션은 `systemd` 유닛의 유형을 서비스 유닛으로 제한합니다. 출력에는 다음과 같은 열이 있습니다.

UNIT

서비스 유닛 이름입니다.

LOAD

`systemd` 데몬이 유닛 구성을 정확하게 구문 분석하고 유닛을 메모리에 로드했는지 여부입니다.

ACTIVE

유닛의 개괄적인 활성화 상태입니다. 이 정보는 유닛이 성공적으로 시작되었는지 여부를 나타냅니다.

SUB

유닛의 구체적인 활성화 상태입니다. 이 정보는 유닛에 대한 보다 자세한 정보를 나타냅니다. 정보는 유닛 유형, 상태 및 유닛 실행 방법에 따라 다릅니다.

DESCRIPTION

유닛에 대한 간단한 설명입니다.

기본적으로 `systemctl list-units --type=service` 명령은 활성화 상태가 `active`인 서비스 유닛만 나열합니다. `systemctl list-units --all` 옵션은 활성화 상태와 관계없이 모든 서비스 유닛을 표시합니다. `--state=` 옵션을 사용하여 `LOAD`, `ACTIVE` 또는 `SUB` 필드의 값을 기준으로 필터링합니다.

```
[root@host ~]# systemctl list-units --type=service --all
UNIT                  LOAD    ACTIVE   SUB      DESCRIPTION
atd.service           loaded  active   running  Job spooling tools
auditd.service        loaded  active   running  Security Auditing ...
auth-rpcgss-module.service loaded  inactive  dead     Kernel Module ...
chronyd.service       loaded  active   running  NTP client/server
cpupower.service      loaded  inactive  dead     Configure CPU power ...
crond.service         loaded  active   running  Command Scheduler
dbus.service          loaded  active   running  D-Bus System Message Bus
● display-manager.service not-found inactive  dead     display-manager.service
...output omitted...
```

인수 없이 `systemctl` 명령을 실행하면 로드되고 활성화된 유닛이 표시됩니다.

```
[root@host ~]# systemctl
UNIT                                     LOAD ACTIVE SUB   DESCRIPTION
proc-sys-fs-binfmt_misc.automount      loaded active waiting  Arbitrary...
sys-devices-....device                loaded active plugged  Virtio network...
sys-subsystem-net-devices-ens3.device loaded active plugged  Virtio network...
...output omitted...
-.mount                                  loaded active mounted  Root Mount
boot.mount                             loaded active mounted  /boot
...output omitted...
systemd-ask-password-plymouth.path    loaded active waiting  Forward Password...
systemd-ask-password-wall.path       loaded active waiting  Forward Password...
init.scope                            loaded active running System and Servi...
session-1.scope                      loaded active running Session 1 of...
atd.service                           loaded active running Job spooling tools
audited.service                      loaded active running Security Auditing...
chronyd.service                     loaded active running NTP client/server
crond.service                        loaded active running Command Scheduler
...output omitted...
```

`systemctl` 명령의 `list-units` 옵션은 `systemd` 서비스가 구문 분석하여 메모리에 로드하려는 유닛을 표시합니다. 이 옵션은 설치되었지만 활성화되지 않은 서비스를 표시하지 않습니다. `systemctl` 명령의 `list-unit-files` 옵션을 사용하여 설치된 모든 유닛 파일의 상태를 확인할 수 있습니다.

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                STATE   VENDOR PRESET
arp-ethers.service                         disabled
atd.service                               enabled
audited.service                          enabled
auth-rpcgss-module.service               static
autovt@.service                          alias
blk-availability.service                 disabled
...output omitted...
```

`systemctl list-unit-files` 명령의 출력에서 `STATE` 필드에 일반적인 몇 가지 항목은 `enabled`, `disabled`, `static`, `masked`입니다. 모든 `STATE` 값은 `systemctl` 명령 도움말 페이지에 나와 있습니다.

서비스 상태 보기

`systemctl status name.type` 명령을 사용하여 유닛 상태를 봅니다. 유닛 유형이 생략된 경우 명령은 해당 이름의 서비스 유닛을 예상합니다.

```
[root@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
   Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
   Active: active (running) since Mon 2022-03-14 05:38:12 EDT; 25min ago
     Docs: man:sshd(8)
           man:sshd_config(5)
   Main PID: 1114 (sshd)
      Tasks: 1 (limit: 35578)
     Memory: 5.2M
        CPU: 64ms
```

```
CGroup: /system.slice/sshd.service
└─1114 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 14 05:38:12 workstation systemd[1]: Starting OpenSSH server daemon...
Mar 14 05:38:12 workstation sshd[1114]: Server listening on 0.0.0.0 port 22.
Mar 14 05:38:12 workstation sshd[1114]: Server listening on :: port 22.
Mar 14 05:38:12 workstation systemd[1]: Started OpenSSH server daemon.
...output omitted...
```

`systemctl` 명령의 `status` 옵션 출력에 표시되는 몇 가지 필드는 다음과 같습니다.

서비스 유닛 정보

필드	설명
로드됨	서비스 유닛이 메모리에 로드되었는지 여부입니다
활성화	서비스 유닛이 실행되고 있는지와 실행 중인 경우 실행된 기간입니다
Docs	서비스에 대한 자세한 정보를 찾을 수 있는 위치입니다
기본 PID	명령 이름을 포함하여 서비스의 기본 프로세스 ID입니다
상태	서비스에 대한 자세한 정보입니다
프로세스	관련 프로세스에 대한 자세한 정보입니다
CGroup	관련 제어 그룹에 대한 자세한 정보입니다

이러한 모든 필드가 항상 명령 출력에 표시되는 것은 아닙니다.

상태 출력의 키워드는 서비스 상태를 나타냅니다.

systemctl 출력의 서비스 상태

키워드	설명
로드됨	유닛 구성 파일이 처리되었습니다.
활성(실행 중)	프로세스가 계속되면서 서비스가 실행 중입니다.
활성(종료됨)	서비스가 일회성 구성을 성공적으로 완료했습니다.
활성(대기 중)	서비스가 실행 중이지만 이벤트를 기다리고 있습니다.
비활성	서비스가 실행되고 있지 않습니다.
활성화됨	서비스가 부팅할 때 시작됩니다.
비활성화됨	서비스가 부팅할 때 시작되도록 설정되어 있지 않습니다.
정적	서비스를 활성화할 수 없지만, 활성화된 유닛에서 서비스를 자동으로 시작할 수 있습니다.

**참고**

`systemctl status NAME` 명령은 Red Hat Enterprise Linux 6 및 이전 버전의 `service NAME status` 명령을 대체합니다.

서비스 상태 확인

`systemctl` 명령은 특정 서비스 상태를 확인합니다. 예를 들어 `systemctl` 명령의 `is-active` 옵션을 사용하여 서비스 유닛이 활성 상태(실행 중)인지 확인합니다.

```
[root@host ~]# systemctl is-active sshd.service
active
```

이 명령은 서비스 유닛의 상태를 반환하며, 일반적으로 `active` 또는 `inactive`입니다.

`systemctl` 명령의 `is-enabled` 옵션을 실행하여 시스템을 부팅할 때 자동으로 시작되도록 서비스 유닛이 활성화되었는지 여부를 확인합니다.

```
[root@host ~]# systemctl is-enabled sshd.service
enabled
```

이 명령은 부팅할 때 시작되도록 서비스 유닛이 활성화되었는지 여부를 반환하며, 일반적으로 상태는 `enabled` 또는 `disabled`입니다.

시작 중에 유닛이 실패했는지 여부를 확인하려면 `systemctl` 명령의 `is-failed` 옵션을 실행합니다.

```
[root@host ~]# systemctl is-failed sshd.service
active
```

이 명령은 서비스가 정확히 실행 중인 경우 `active`, 시작 중에 오류가 발생한 경우 `failed`를 반환합니다. 유닛이 중지된 경우에는 `unknown` 또는 `inactive`를 반환합니다.

실패한 유닛을 모두 표시하려면 `systemctl --failed --type=service` 명령을 실행합니다.

**참조**

`systemd(1)`, `systemd.unit(5)`, `systemd.service(5)`, `systemd.socket(5)` 및 `systemctl(1)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#managing-services-with-systemd_configuring-basic-system-settings

에서 Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide의 Managing Services with systemd 장을 참조하십시오.

▶ 연습 가이드

자동으로 시작된 시스템 프로세스 식별

이 연습에서는 설치된 서비스 유닛을 표시하고, 현재 활성화되어 서버에서 활성 상태인 서비스를 식별합니다.

결과

- 설치된 서비스 유닛을 표시합니다.
- 시스템에서 활성화되어 활성 상태인 서비스를 식별합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start services-identify
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
[student@servera ~]$
```

- ▶ 2. **servera** 시스템에 설치된 서비스 유닛을 모두 표시합니다.

```
[student@servera ~]$ systemctl list-units --type=service
UNIT                  LOAD     ACTIVE    SUB      DESCRIPTION
atd.service           loaded   active   running  Deferred execution scheduler
auditd.service        loaded   active   running  Security Auditing Service
chronyd.service       loaded   active   running  NTP client/server
crond.service         loaded   active   running  Command Scheduler
dbus-broker.service   loaded   active   running  D-Bus System Message Bus
...output omitted...
```

q 를 눌러 명령을 종료합니다.

- ▶ 3. **servera** 시스템의 활성 및 비활성 소켓 유닛을 모두 표시합니다.

```
[student@servera ~]$ systemctl list-units --type=socket --all
UNIT                  LOAD     ACTIVE    SUB      DESCRIPTION
dbus.socket           loaded   active   running  D-Bus System Message Bus Socket
dm-event.socket       loaded   active   listening Device-mapper event daemon FIFOs
lvm2-lvmpolld.socket loaded   active   listening LVM2 poll daemon socket
...output omitted...
```

```

LOAD  = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB   = The low-level unit activation state, values depend on unit type.
13 loaded units listed.

To show all installed unit files use 'systemctl list-unit-files'.

```

- ▶ 4. **chronyd** 서비스의 상태를 확인합니다. NTP(네트워크 시간 프로토콜) 동기화에 이 서비스를 사용할 수 있습니다.

4.1. **chronyd** 서비스의 상태를 표시합니다. 활성 데몬의 프로세스 ID를 확인합니다.

```

[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
  preset: enabled)
    Active: active (running) since Mon 2022-03-14 05:38:15 EDT; 1h 16min ago
      Docs: man:chronyd(8)
             man:chrony.conf(5)
  Process: 728 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/
SUCCESS)
 Main PID: 747 (chronyd)
   Tasks: 1 (limit: 10800)
     Memory: 3.7M
        CPU: 37ms
      CGroup: /system.slice/chronyd.service
              └─747 /usr/sbin/chronyd -F 2

Mar 14 05:38:15 servera.lab.example.com systemd[1]: Starting NTP client/server...
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: chronyd version 4.1 starting
(+CMDMON +NTP +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH
+IPV6 +DEBUG)
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: commandkey directive is no
longer supported
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: generatecommandkey directive
is no longer supported
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: Frequency -11.870 +/- 1.025
ppm read from /var/lib/chrony/drift
Mar 14 05:38:15 servera.lab.example.com chronyd[747]: Loaded seccomp filter (level
2)
Mar 14 05:38:15 servera.lab.example.com systemd[1]: Started NTP client/server.
Mar 14 05:38:23 servera.lab.example.com chronyd[747]: Selected source
172.25.254.254

```

q를 눌러 명령을 종료합니다.

- 4.2. 프로세스 ID를 사용하여 **chronyd** 데몬이 실행 중인지 확인합니다. 이전 명령에서 **chronyd** 서비스와 연결된 프로세스 ID의 출력은 747입니다. 사용하는 시스템의 프로세스 ID는 이 값과 다를 수 있습니다.

```

[student@servera ~]$ ps -p 747
  PID TTY          TIME CMD
 747 ?        00:00:00 chronyd

```

- ▶ 5. **sshd** 서비스의 상태를 확인합니다. 시스템 간의 암호화된 보안 통신을 위해 이 서비스를 사용할 수 있습니다.

5.1. 시스템 부팅 시 시작되도록 **sshd** 서비스가 설정되었는지 여부를 확인합니다.

```
[student@servera ~]$ systemctl is-enabled sshd
enabled
```

5.2. 모든 상태 정보를 표시하지 않고 **sshd** 서비스가 활성 상태인지 확인합니다.

```
[student@servera ~]$ systemctl is-active sshd
active
```

5.3. **sshd** 서비스의 상태를 표시합니다.

```
[student@servera ~]$ systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Mon 2022-03-14 05:38:16 EDT; 1h 19min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 784 (sshd)
     Tasks: 1 (limit: 10800)
   Memory: 6.7M
      CPU: 82ms
     CGroup: /system.slice/sshd.service
             └─784 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 14 05:38:16 servera.lab.example.com systemd[1]: Starting OpenSSH server
daemons...
Mar 14 05:38:16 servera.lab.example.com sshd[784]: Server listening on 0.0.0.0
port 22.
Mar 14 05:38:16 servera.lab.example.com sshd[784]: Server listening on :: port 22.
Mar 14 05:38:16 servera.lab.example.com systemd[1]: Started OpenSSH server daemon.
Mar 14 06:51:36 servera.lab.example.com sshd[1090]: Accepted publickey for student
from 172.25.250.9 port 53816 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFCT
+wowZLNzNlBT0
Mar 14 06:51:36 servera.lab.example.com sshd[1090]: pam_unix(sshd:session):
  session opened for user student(uid=1000) by (uid=0)
```

q를 눌러 명령을 종료합니다.

- ▶ 6. 모든 서비스 유닛의 활성화 또는 비활성화 상태를 나열합니다.

```
[student@servera ~]$ systemctl list-unit-files --type=service
UNIT FILE                           STATE        VENDOR PRESET
arp-ethers.service                  disabled    disabled
atd.service                         enabled     enabled
auditd.service                      enabled     enabled
auth-rpcgss-module.service          static      -
autovt@.service                     alias      -
blk-availability.service           disabled    disabled
```

```
bluetooth.service           enabled      enabled
chrony-wait.service        disabled     disabled
chronyd.service            enabled      enabled
...output omitted...
```

q 를 눌러 명령을 종료합니다.

▶ 7. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish services-identify
```

이것으로 섹션을 완료합니다.

시스템 서비스 제어

목표

`systemctl`을 사용하여 시스템 데몬 및 네트워크 서비스를 제어합니다.

서비스 시작 및 중지

수동으로 서비스를 시작, 중지 또는 다시 로드하여 서비스를 업데이트하거나, 구성 파일을 업데이트하거나, 서비스를 제거하거나, 자주 사용하지 않는 서비스를 수동으로 관리할 수 있습니다.

`systemctl status` 명령을 사용하여 서비스가 실행 중인지 또는 중지되었는지 서비스 상태를 확인합니다.

```
[root@host ~]# systemctl status sshd.service
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Wed 2022-03-23 11:58:18 EDT; 2min 56s ago
    ...output omitted...
```

root 사용자로 `systemctl start` 명령을 사용합니다(필요한 경우 `sudo` 명령 사용). 서비스 유형 없이 서비스 이름만 사용하여 `systemctl start` 명령을 실행하면 `systemd` 서비스에서 `.service` 파일을 찾습니다.

```
[root@host ~]# systemctl start sshd
```

실행 중인 서비스를 중지하려면 `systemctl` 명령의 `stop` 옵션을 사용합니다. 다음 예제에서는 `sshd.service` 서비스를 중지하는 방법을 보여줍니다.

```
[root@host ~]# systemctl stop sshd.service
```

서비스 재시작 및 다시 로드

실행 중인 서비스를 재시작하면 서비스가 먼저 중지된 후 다시 시작됩니다. 서비스 재시작 시 새 프로세스는 시작 중에 새 ID를 받게 되므로 프로세스 ID가 변경됩니다. 실행 중인 서비스를 재시작하려면 `systemctl` 명령의 `restart` 옵션을 사용합니다. 다음 예제에서는 `sshd` 서비스를 재시작하는 방법을 보여줍니다.

```
[root@host ~]# systemctl restart sshd.service
```

일부 서비스는 재시작하지 않고도 구성 파일을 다시 로드할 수 있습니다(서비스 다시 로드라고 함). 서비스를 다시 로드하는 경우 다양한 서비스 프로세스와 연결된 프로세스 ID가 변경되지 않습니다. 실행 중인 서비스를 다시 로드하려면 `systemctl` 명령의 `reload` 옵션을 사용합니다. 다음 예제에서는 구성이 변경된 후 `sshd.service` 서비스를 다시 로드하는 방법을 보여줍니다.

```
[root@host ~]# systemctl reload sshd.service
```

서비스에 구성 파일 변경 사항을 다시 로드하는 기능이 있는지 잘 모르겠으면 **systemctl** 명령의 **reload-or-restart** 인수를 사용합니다. 이 명령은 다시 로드 기능을 사용할 수 있는 경우 구성 변경 사항을 다시 로드합니다. 해당 기능이 없는 경우에는 명령이 서비스를 다시 시작하여 새로운 구성 변경 사항을 구현합니다.

```
[root@host ~]# systemctl reload-or-restart sshd.service
```

유닛 종속성 표시

일부 서비스는 다른 서비스가 먼저 실행되어야 하며, 이로 인해 다른 서비스에 대한 종속성이 생성됩니다. 다른 서비스는 부팅할 때 시작되지 않고 요청 시에만 시작됩니다. 두 경우 모두, **systemd** 및 **systemctl** 명령은 종속성을 해결하기 위해서는, 자주 사용하지 않는 서비스를 시작하기 위해서는 필요에 따라 서비스를 시작합니다. 예를 들어 출력 시스템(CUPS) 서비스가 실행되고 있지 않은데 출력 스플 디렉터리에 파일을 배치하면 시스템이 CUPS 관련 데몬이나 명령을 시작하여 출력 요청을 충족합니다.

```
[root@host ~]# systemctl stop cups.service
Warning: Stopping cups, but it can still be activated by:
  cups.path
  cups.socket
```

그러나 시스템에서 출력 서비스를 중지하려면 3개 유닛을 모두 중지해야 합니다. 서비스를 비활성화하면 종속성도 비활성화됩니다.

systemctl list-dependencies UNIT 명령은 서비스 유닛을 시작하는 종속성의 계층 구조 매핑을 표시합니다. 역방향 종속성(지정한 유닛에 종속된 유닛)을 표시하려면 명령에 **--reverse** 옵션을 사용합니다.

```
[root@host ~]# systemctl list-dependencies sshd.service
sshd.service
• └─system.slice
•   └─ssh-keygen.target
•     └─ssh-keygen@ecdsa.service
•     └─ssh-keygen@ed25519.service
•     └─ssh-keygen@rsa.service
•   └─sysinit.target
...output omitted...
```

서비스 마스킹 및 마스킹 해제

시스템에 설치된 서비스 간에 충돌이 발생하는 경우도 있습니다. 예를 들어 메일 서버(**postfix** 및 **sendmail** 서비스)를 관리하는 방법에는 여러 가지가 있습니다. 서비스를 마스킹하면 관리자가 다른 서비스와 충돌하는 서비스를 실수로 시작하지 않도록 방지할 수 있습니다. 마스킹하면 구성 디렉터리에 서비스가 시작되지 않도록 하는 **/dev/null** 파일에 대한 링크가 생성됩니다. 서비스를 마스킹하려면 **systemctl** 명령의 **mask** 옵션을 사용합니다.

```
[root@host ~]# systemctl mask sendmail.service
Created symlink /etc/systemd/system/sendmail.service → /dev/null.
```

그런 다음 **systemctl list-unit-files** 명령을 사용하여 서비스 상태를 확인합니다.

```
[root@host ~]# systemctl list-unit-files --type=service
UNIT FILE                                     STATE
...output omitted...
sendmail.service                               masked
...output omitted...
```

마스킹된 서비스 유닛을 시작하려고 하면 실패하고 다음과 같은 내용이 출력됩니다.

```
[root@host ~]# systemctl start sendmail.service
Failed to start sendmail.service: Unit sendmail.service is masked.
```

서비스 유닛의 마스킹을 해제하려면 `systemctl unmask` 명령을 사용합니다.

```
[root@host ~]# systemctl unmask sendmail
Removed /etc/systemd/system/sendmail.service.
```



중요

비활성화된 서비스를 수동이나 다른 유닛 파일을 통해 시작할 수 있지만, 부팅 시 자동으로 시작되는 않습니다. 마스킹된 서비스는 수동 또는 자동으로 시작되지 않습니다.

부팅할 때 시작 또는 중지되도록 서비스 활성화

실행 중인 시스템에서 서비스를 시작하는 경우 시스템이 재부팅될 때 서비스가 자동으로 시작된다는 보장이 없습니다. 마찬가지로, 실행 중인 시스템에서 서비스를 중지하는 경우 시스템이 재부팅될 때 다시 시작되지 않는다는 보장이 없습니다. `systemd` 구성 디렉터리에 링크를 생성하면 부팅 시 서비스가 시작됩니다. `systemctl` 명령에 `enable` 또는 `disable` 옵션을 사용하여 이러한 링크를 생성하거나 제거할 수 있습니다.

```
[root@root ~]# systemctl enable sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

이 명령은 일반적으로 `/usr/lib/systemd/system` 디렉터리의 서비스 유닛 파일과 `systemd` 명령이 `/etc/systemd/system/TARGETNAME.target.wants` 디렉터리에서 파일을 찾는 디스크 위치 간의 심볼릭 링크를 생성합니다. 서비스를 활성화하는 경우 현재 세션에서는 서비스가 시작되지 않습니다. 서비스를 시작하고 부팅 시 자동으로 시작되도록 활성화하려면 `systemctl start` 및 `systemctl enable` 명령을 둘 다 실행하거나 동등한 `systemctl enable --now` 명령을 사용합니다.

```
[root@root ~]# systemctl enable --now sshd.service
Created symlink /etc/systemd/system/multi-user.target.wants/sshd.service → /usr/
lib/systemd/system/sshd.service.
```

서비스가 자동으로 시작되지 않도록 비활성화하려면 `systemctl disable` 명령을 사용합니다. 이 명령을 실행하면 서비스를 활성화하는 동안 생성된 심볼릭 링크가 제거됩니다. 서비스를 비활성화해도 현재 실행 중인 서비스는 중지되지 않습니다. 서비스를 비활성화하고 중지하려면 `systemctl stop` 및 `systemctl disable` 명령을 둘 다 실행하거나 동등한 `systemctl disable --now` 명령을 사용합니다.

```
[root@host ~]# systemctl disable --now sshd.service
Removed /etc/systemd/system/multi-user.target.wants/sshd.service.
```

서비스가 활성화 또는 비활성화되었는지 확인하려면 `systemctl is-enabled` 명령을 사용합니다.

```
[root@host ~]# systemctl is-enabled sshd.service
enabled
```

systemctl 명령 요약

실행 중인 시스템에서 서비스를 시작 및 중지할 수 있으며, 부팅 시 자동으로 시작되도록 활성화하거나 비활성화할 수 있습니다.

유용한 서비스 관리 명령

명령	작업
<code>systemctl status UNIT</code>	유닛 상태에 대한 세부 정보를 봅니다.
<code>systemctl stop UNIT</code>	실행 중인 시스템에서 서비스를 중지합니다.
<code>systemctl start UNIT</code>	실행 중인 시스템에서 서비스를 시작합니다.
<code>systemctl restart UNIT</code>	실행 중인 시스템에서 서비스를 다시 시작합니다.
<code>systemctl reload UNIT</code>	실행 중인 서비스의 구성 파일을 다시 로드합니다.
<code>systemctl mask UNIT</code>	수동으로 및 부팅할 때 시작되지 않도록 서비스를 비활성화합니다.
<code>systemctl unmask UNIT</code>	마스킹된 서비스를 사용 가능하도록 만듭니다.
<code>systemctl enable UNIT</code>	부팅할 때 서비스가 시작되도록 구성합니다. <code>--now</code> 옵션을 사용하여 서비스를 시작할 수도 있습니다.
<code>systemctl disable UNIT</code>	부팅할 때 서비스가 시작되지 않도록 비활성화합니다. <code>--now</code> 옵션을 사용하여 서비스를 종지할 수도 있습니다.



참조

**systemd(1), systemd.unit(5), systemd.service(5), systemd.socket(5) 및
systemctl(1)** 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-system-services-with-systemctl_configuring-basic-system-settings

에서 Red Hat Enterprise Linux 9 Configuring Basic System Settings Guide의 Managing System Services with systemctl 장을 참조하십시오.

▶ 연습 가이드

시스템 서비스 제어

이 연습에서는 **systemctl** 명령을 사용하여 **systemd** 서비스를 중지, 시작, 재시작, 다시 로드, 활성화 및 비활성화합니다.

결과

- systemctl** 명령을 사용하여 **systemd** 서비스를 제어합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start services-control
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **sshd** 서비스를 재시작하고 다시 로드한 다음, 결과를 관찰합니다.

- 2.1. **sshd** 서비스의 상태를 표시합니다. **sshd** 데몬의 프로세스 ID를 확인합니다. **q** 를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:04:45 EDT; 16min ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 784 (sshd)
     Tasks: 1 (limit: 10799)
    Memory: 6.6M
  ...output omitted...
```

- 2.2. **sshd** 서비스를 다시 시작하고 상태를 봅니다. 이 예제에서는 데몬의 프로세스 ID가 784에서 1193으로 변경됩니다. **q** 를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl restart sshd
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:21:40 EDT; 5s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Main PID: 1193 (sshd)
    Tasks: 1 (limit: 10799)
   Memory: 1.7M
  ...output omitted...
```

- 2.3. **sshd** 서비스를 재로드하고 상태를 봅니다. 데몬의 프로세스 ID가 변경되지 않습니다. **q**를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl reload sshd
[root@servera ~]# systemctl status sshd
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:21:40 EDT; 52s ago
    Docs: man:sshd(8)
          man:sshd_config(5)
  Process: 1201 ExecReload=/bin/kill -HUP $MAINPID (code=exited, status=0/
 SUCCESS)
  Main PID: 1193 (sshd)
    Tasks: 1 (limit: 10799)
   Memory: 1.7M
  ...output omitted...
```

- ▶ 3. **chronyd** 서비스가 실행 중인지 확인합니다. **q**를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
 preset: enabled)
  Active: active (running) since Thu 2022-05-19 04:04:44 EDT; 19min ago
  ...output omitted...
```

- ▶ 4. **chronyd** 서비스를 중지하고 상태를 봅니다. **q**를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl stop chronyd
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
 preset: enabled)
  Active: inactive (dead) since Thu 2022-05-19 04:24:59 EDT; 4s ago
  ...output omitted...
```

```
May 19 04:24:59 servera.lab.example.com systemd[1]: Stopping NTP client/server...
May 19 04:24:59 servera.lab.example.com systemd[1]: chronyd.service: Deactivated
successfully.
May 19 04:24:59 servera.lab.example.com systemd[1]: Stopped NTP client/server.
```

- ▶ 5. 시스템 부팅 시 시작되도록 **chronyd** 서비스가 설정되었는지 여부를 확인합니다.

```
[root@servera ~]# systemctl is-enabled chronyd
enabled
```

- ▶ 6. **servera** 시스템을 재부팅한 다음, **chronyd** 서비스의 상태를 봅니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

student 사용자로 **servera** 시스템에 로그인한 후 **root** 사용자로 전환합니다. **chronyd** 서비스의 상태를 봅니다. **q** 를 눌러 명령을 종료합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
    Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor
    preset: enabled)
      Active: active (running) since Thu 2022-05-19 04:27:12 EDT; 40s ago
        ...output omitted...
```

- ▶ 7. 부팅할 때 시작되지 않도록 **chronyd** 서비스를 비활성화한 다음, 서비스 상태를 봅니다. **q** 를 눌러 명령을 종료합니다.

```
[root@servera ~]# systemctl disable chronyd
Removed /etc/systemd/system/multi-user.target.wants/chronyd.service.
[root@servera ~]# systemctl status chronyd
● chronyd.service - NTP client/server
    Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
    preset: enabled)
      Active: active (running) since Thu 2022-05-19 04:27:12 EDT; 2min 48s ago
        ...output omitted...
```

▶ 8. `servera`를 재부팅한 다음, `chronyd` 서비스의 상태를 봅니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

`student` 사용자로 `servera`에 로그인하고 `chronyd` 서비스의 상태를 봅니다. `q` 를 눌러 명령을 종료합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ systemctl status chronyd
● chronyd.service - NTP client/server
  Loaded: loaded (/usr/lib/systemd/system/chronyd.service; disabled; vendor
  preset: enabled)
    Active: inactive (dead)
      Docs: man:chronyd(8)
            man:chrony.conf(5)
```

▶ 9. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish services-control
```

이것으로 섹션을 완료합니다.

부팅 타겟 선택

목표

Red Hat Enterprise Linux 부팅 프로세스를 설명하고, 부팅 시 기본 타겟을 설정하고, 시스템을 기본값이 아닌 타겟으로 부팅합니다.

Red Hat Enterprise Linux 9 부팅 프로세스를 설명합니다.

오늘날의 컴퓨터 시스템은 하드웨어와 소프트웨어가 복잡하게 결합되어 있습니다. 정의되지 않은 상태 즉 전원이 꺼진 상태에서 로그인 프롬프트를 사용하는 실행 시스템으로 시작하려면 많은 하드웨어와 소프트웨어가 연동해야 합니다. 다음 목록에는 Red Hat Enterprise Linux 9을 부팅하는 실제 **x86_64** 시스템과 관련된 작업이 요약되어 있습니다. **x86_64** 가상 시스템에 대한 목록은 거의 동일하지만 하이퍼바이저가 소프트웨어의 일부 하드웨어별 단계를 처리합니다.

- 시스템 전원을 켭니다. 시스템 펌웨어(최신 UEFI 또는 기존 BIOS)가 POST(Power On Self Test)를 실행한 다음 하드웨어를 초기화하기 시작합니다.

시스템 BIOS 또는 UEFI는 부팅 절차 초기에 특정 키 조합(**F2** 등)을 눌러 구성합니다.

- UEFI 부트 펌웨어는 모든 디스크에서 MBR(Master Boot Record)을 검색하거나 구성하는 부팅 가능 장치를 검색함으로써 구성합니다.

시스템 BIOS 또는 UEFI 구성은 부팅 절차 초기에 특정 키 조합(**F2** 등)을 눌러 구성합니다.

- 시스템 펌웨어가 디스크에서 부트 로더를 읽은 다음 시스템 제어 권한을 해당 부트 로더로 전달합니다. Red Hat Enterprise Linux 9 시스템의 부트 로더는 GRand Unified Bootloader 버전 2(GRUB2)입니다.

grub2-install 명령은 GRUB2를 BIOS 시스템용 디스크에 부트 로더로 설치합니다. **grub2-install** 명령을 직접 사용하여 UEFI 부트 로더를 설치하지 마십시오. RHEL 9에서는 사전 빌드된 **/boot/efi/EFI/redhat/grubx64.efi** 파일을 제공하는데 이 파일에는 보안 부팅 시스템에 필요한 인증 서명이 포함되어 있습니다. UEFI 시스템에서 직접 **grub2-install**을 실행하면 이러한 필수 서명 없이 새 **grubx64.efi** 파일이 생성됩니다. **grub2-efi** 패키지에서 올바른 **grubx64.efi** 파일을 복원할 수 있습니다.

- GRUB2는 BIOS의 경우 **/boot/grub2/grub.cfg** 파일에서, UEFI의 경우 **/boot/efi/EFI/redhat/grub.cfg** 파일에서 해당 구성을 로드하고, 부팅할 커널을 선택할 수 있는 메뉴를 표시합니다.

GRUB2는 **/etc/grub.d/** 디렉터리 및 **/etc/default/grub** 파일을 사용하여 구성됩니다.

grub2-mkconfig 명령은 BIOS 또는 UEFI에 대해 각각 **/boot/grub2/grub.cfg** 및 **/boot/efi/EFI/redhat/grub.cfg** 파일을 생성합니다.

- 커널을 선택하거나 제한 시간이 만료되면 부트 로더가 디스크에서 커널 및 initramfs를 로드하여 메모리에 배치합니다. **initramfs** 이미지는 부팅, 초기화 스크립트 등에 필요한 모든 하드웨어에 대한 kernel 모듈이 포함된 아카이브입니다. Red Hat Enterprise Linux 9에서 **initramfs** 이미지에는 실행 중인 커널과 **systemd** 유닛이 있는 부팅 가능한 루트 파일 시스템이 포함되어 있습니다.

initramfs 이미지는 **/etc/dracut.conf.d/** 디렉터리, **dracut** 명령 및 **initramfs** 파일을 검사하는 **lsinitrd** 명령을 사용하여 구성합니다.

- 부트 로더는 제어 권한을 커널로 전달하여 부트 로더의 커널 명령줄에 지정된 옵션과 메모리의 **initramfs** 이미지 위치를 전달합니다.

부트 로더는 `/etc/grub.d/` 디렉터리, `/etc/default/grub` 파일 및 `/boot/grub2/grub.cfg` 파일을 생성하는 `grub2-mkconfig` 명령을 사용하여 구성합니다.

- 커널은 `initramfs` 이미지에서 드라이버를 찾을 수 있는 모든 하드웨어를 초기화한 다음 `initramfs` 이미지에서 PID 1로 `/sbin/init` 스크립트를 실행합니다. Red Hat Enterprise Linux 9에서 `/sbin/init` 스크립트는 `systemd` 유닛에 대한 링크입니다.

스크립트는 커널 `init=` 명령줄 매개 변수를 사용하여 구성합니다.

- `initramfs` 이미지의 `systemd` 유닛은 `initrd.target` 대상에 대한 모든 유닛을 실행합니다. 이 유닛에는 루트 파일 시스템을 디스크의 `/sysroot` 디렉터리에 마운트하는 작업이 포함됩니다.

`/etc/fstab` 파일을 사용하여 구성됩니다.

- 커널은 루트 파일 시스템을 `initramfs` 이미지에서 `/sysroot` 디렉터리의 루트 파일 시스템으로 전환(피벗)합니다. 그러면 `systemd` 유닛은 디스크에 설치된 `systemd` 복사본을 사용하여 자체적으로 다시 실행됩니다.

- `systemd` 유닛은 커널 명령줄에서 전달되거나 시스템에 구성된 기본 타겟을 찾습니다. 그런 다음 `systemd` 유닛은 해당 대상의 구성을 준수하도록 유닛을 시작(및 중지)하고, 유닛 간 종속성을 자동으로 해결합니다. `systemd` 유닛은 시스템이 의도한 상태에 도달하기 위해 활성화하는 유닛 집합입니다. 이러한 대상에서는 일반적으로 텍스트 기반 로그인 또는 그래픽 로그인 화면이 시작됩니다.

`/etc/systemd/system/default.target` 파일 및 `/etc/systemd/system/` 디렉터리를 사용하여 구성됩니다.

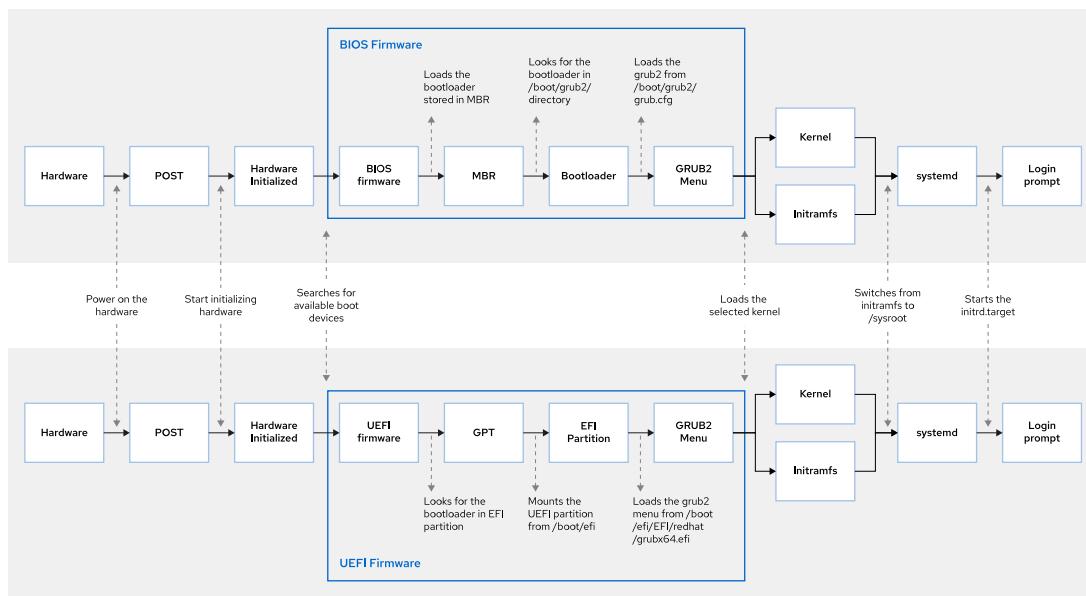


그림 11.1: BIOS 기반 및 UEFI 기반 시스템의 부팅 프로세스

전원 끄기 및 재부팅

`systemctl` 명령을 사용하면 명령줄에서 실행 중인 시스템 전원을 끄거나 재부팅할 수 있습니다.

`systemctl poweroff` 명령은 실행 중인 모든 서비스를 중단하고 모든 파일 시스템을 마운트 해제한다음(또는 마운트 해제할 수 없는 경우 읽기 전용으로 다시 마운트) 시스템 전원을 끕니다.

systemctl reboot 명령은 실행 중인 모든 서비스를 종단하고 모든 파일 시스템을 마운트 해제한 다음 시스템을 재부팅합니다.

이러한 명령의 짧은 버전인 **poweroff** 및 **reboot**를 사용할 수도 있는데, 이는 해당 **systemctl** 항목에 대한 심볼릭 링크입니다.



참고

systemctl halt 및 **halt** 명령을 사용하여 시스템을 중지할 수도 있습니다. 이러한 명령은 **poweroff** 명령과 달리 시스템 전원을 끄지 않고 전원을 수동으로 안전하게 끌 수 있는 지점으로 시스템을 전환합니다.

Systemd 타겟 선택

systemd 타겟은 원하는 상태에 도달하기 위해 시작해야 하는 **systemd** 유닛 집합입니다. 다음 표에는 가장 중요한 대상이 나열되어 있습니다.

일반적으로 사용되는 대상

타겟	목적
graphical.target	이 대상은 멀티 유저를 지원하고, 그래픽 및 텍스트 기반 로그인을 제공합니다.
multi-user.target	이 대상은 멀티 유저를 지원하고, 텍스트 기반 로그인을 제공합니다.
rescue.target	이 대상은 시스템을 복구할 수 있는 단일 사용자 시스템을 제공합니다.
emergency.target	이 대상은 rescue.target 장치가 시작되지 않을 때 시스템을 복구하기 위해 가장 최소한의 시스템을 시작합니다.

대상은 다른 대상의 일부일 수 있습니다. 예를 들어, **graphical.target** 유닛에는 **multi-user.target** 유닛이 포함되며, 이 유닛은 **basic.target** 유닛 및 기타 유닛에 종속됩니다. 다음 명령을 사용하여 이러한 종속성을 볼 수 있습니다.

```
[user@host ~]$ systemctl list-dependencies graphical.target | grep target
graphical.target
* └─multi-user.target
*   ├─basic.target
*   | ├─paths.target
*   | ├─slices.target
*   | ├─sockets.target
*   | ├─sysinit.target
*   | | ├─cryptsetup.target
*   | | ├─integritysetup.target
*   | | ├─local-fs.target
...output omitted...
```

사용 가능한 대상을 나열하려면 다음 명령을 사용하십시오.

```
[user@host ~]$ systemctl list-units --type=target --all
UNIT                  LOAD     ACTIVE   SUB      DESCRIPTION
...
basic.target          loaded   active   active  Basic System
...output omitted...
cloud-config.target  loaded   active   active  Cloud-config availability
cloud-init.target    loaded   active   active  Cloud-init target
cryptsetup-pre.target loaded   inactive dead    Local Encrypted Volumes
(Pre)
cryptsetup.target    loaded   active   active  Local Encrypted Volumes
...output omitted...
```

런타임 시 타겟 선택

관리자는 실행 중인 시스템에서 `systemctl isolate` 명령을 사용하여 다른 타겟으로 전환할 수 있습니다.

```
[root@host ~]# systemctl isolate multi-user.target
```

대상을 격리하면 해당 대상(및 해당 종속성)에서 요구하지 않는 모든 서비스가 중지되며 아직 시작되지 않은 모든 필수 서비스가 시작됩니다.

모든 대상을 격리할 수는 없습니다. 유닛 파일에 `AllowIsolate=yes` 설정이 있는 대상만 격리할 수 있습니다. 예를 들어 그래픽 타겟은 격리할 수 있지만 `cryptsetup` 타겟은 격리할 수 없습니다.

```
[user@host ~]$ systemctl cat graphical.target
# /usr/lib/systemd/system/graphical.target
...output omitted...
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.service
AllowIsolate=yes
[user@host ~]$ systemctl cat cryptsetup.target
# /usr/lib/systemd/system/cryptsetup.target
...output omitted...
[Unit]
Description=Local Encrypted Volumes
Documentation=man:systemd.special(7)
```

기본 타겟 설정

시스템이 시작되면 `systemd` 유닛에서 `default.target` 대상을 활성화합니다. 일반적으로 `/etc/systemd/system/` 디렉터리의 기본 타겟은 `graphical.target` 또는 `multi-user.target` 타겟에 대한 심볼릭 링크입니다. 이 심볼릭 링크를 직접 편집하는 대신 `systemctl` 명령에는 이 링크를 관리하는 두 가지 하위 명령, `get-default` 및 `set-default`가 제공됩니다.

```
[root@host ~]# systemctl get-default
multi-user.target
[root@host ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
graphical.target.
[root@host ~]# systemctl get-default
graphical.target
```

부팅 시 다른 타겟 선택

부팅 시 다른 대상을 선택하려면 부트 로더에서 커널 명령줄에 **systemd.unit=target.target** 옵션을 추가합니다.

예를 들어 서비스를 거의 실행하지 않고 시스템 구성은 변경할 수 있는 복구 쉘로 시스템을 부팅하려면 부트 로더의 커널 명령줄에 다음 옵션을 추가합니다.

```
systemd.unit=rescue.target
```

이 구성 변경은 단일 부트에만 영향을 미치므로 부팅 절차 문제를 해결하는 데 유용합니다.

이 방법을 사용하여 다른 타겟을 선택하려면 다음 절차를 사용합니다.

1. 시스템을 부팅 또는 재부팅합니다.
2. 정상적으로 부팅을 시작하는 **Enter** 키를 제외한 임의의 키를 눌러 부트 로더 메뉴 카운트다운을 중단합니다.
3. 커서를 시작할 커널 항목으로 이동합니다.
4. **e** 를 눌러 현재 항목을 편집합니다.
5. 커서를 **linux** 로 시작하는 행으로 이동합니다. 이는 커널 명령줄입니다.
6. **systemd.unit=target.target** 를 추가합니다(예: **systemd.unit=emergency.target**).
7. **Ctrl+x** 를 눌러 이러한 변경 사항을 적용하여 부팅합니다.



참조

info grub2(GNU GRUB manual)

bootup(7), dracut.bootup(7), lsinitrd(1), systemd.target(5),
systemd.special(7), sulogin(8), systemctl(1) 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#managing-services-with-systemd
에서 Managing Services with systemd 가이드의 Configuring Basic System Settings 장을 참조하십시오.

▶ 연습 가이드

부팅 타겟 선택

이 연습에서는 시스템이 부팅되는 기본 타겟을 결정하고 해당 시스템을 다른 타겟으로 부팅합니다.

결과

- 시스템 기본 타겟을 업데이트하고 부트 로더의 임시 타겟을 사용합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start boot-selecting
```

지침

- ▶ 1. **workstation** 시스템에서 터미널을 열고 기본 타겟이 **graphical.target**인지 확인합니다.

```
[student@workstation ~]$ systemctl get-default  
graphical.target
```

- ▶ 2. **workstation** 시스템에서 재부팅하지 않고 **multi-user** 타겟으로 수동 전환합니다. **sudo** 명령을 사용하고 암호를 입력하라는 메시지가 표시되면 암호로 **student**를 사용합니다.

```
[student@workstation ~]$ sudo systemctl isolate multi-user.target  
[sudo] password for student: student
```

- ▶ 3. 텍스트 기반 콘솔에 액세스합니다. 관련 버튼 또는 메뉴 항목을 사용하여 **Ctrl+Alt+F1** 키 시퀀스를 사용합니다. **redhat** 을 암호로 사용하여 **root** 사용자로 로그인합니다.



참고

알림: 웹 페이지를 통해 터미널을 사용하는 경우 웹 브라우저의 URL 표시줄 아래에 있는 화면 오른쪽 메뉴에서 **Show Keyboard** 아이콘을 클릭할 수 있습니다.

```
workstation login: root  
Password: redhat  
[root@workstation ~]#
```

- ▶ 4. 자동으로 **multi-user** 타겟으로 부팅되도록 **workstation** 시스템을 구성한 다음 **workstation** 시스템을 재부팅하여 확인합니다. 부팅이 완료되면 기본 **systemd** 대상을 다시 **graphical** 대상으로 변경합니다.

- 4.1. 기본 타겟을 설정합니다.

```
[root@workstation ~]# systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
multi-user.target.
```

- 4.2. **workstation** 시스템을 재부팅합니다. 재부팅 후에는 시스템에 더 이상 그래픽 로그인 화면이 아닌 텍스트 기반 콘솔이 표시됩니다.

```
[root@workstation ~]# systemctl reboot
```

- 4.3. **root** 사용자로 로그인합니다.

```
workstation login: root
Password: redhat
Last login: Thu Mar 28 14:50:53 on tty1
[root@workstation ~]#
```

- 4.4. 기본 **systemd** 타겟을 다시 **graphical** 타겟으로 설정합니다.

```
[root@workstation ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target -> /usr/lib/systemd/system/
graphical.target.
```

이 단계에서는 기본 **systemd** 타겟 설정을 실습하는 연습의 첫 부분이 완료됩니다.

- ▶ 5. 이 연습의 두 번째 부분에서는 시스템을 복구하는 복구 모드를 사용하여 실습합니다.

workstation을 다시 재부팅하여 부트 로더에 액세스합니다. 부트 로더 메뉴 안에서 **rescue** 대상으로 부팅합니다.

- 5.1. 재부팅을 시작합니다.

```
[root@workstation ~]# systemctl reboot
```

- 5.2. 부트 로더 메뉴가 나타나면 정상적으로 부팅을 시작하는 **Enter** 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.

- 5.3. 커서 키를 사용하여 기본 부트 로더 항목을 강조 표시합니다.

- 5.4. **e** 를 눌러 현재 항목을 편집합니다.

- 5.5. 커서 키를 사용하여 **linux**로 시작하는 행으로 이동합니다.

- 5.6. **End**를 눌러 커서를 행 끝으로 이동합니다.

- 5.7. 행 끝에 **systemd.unit=rescue.target** 을 추가합니다.

**참고**

콘솔에서 텍스트를 읽기 어려운 경우 부트 로더 항목에서 커널 행을 편집할 때 해상도를 변경하는 것이 좋습니다.

콘솔 해상도를 변경하려면 **linux** 단어로 시작하는 행에서 **systemd.unit=rescue.target** 다음에 **video=640x480** 또는 **vga=ask** 를 추가합니다. **video=640x480** 인수를 사용하면 텍스트 콘솔이 약 80열 너비로 표시됩니다. 대신 **vga=ask** 를 사용하면 부팅 시 환경에 가장 적합한 해상도를 선택할 수 있습니다.

5.8. **Ctrl+x** 를 눌러 수정된 구성으로 부팅합니다.

5.9. 복구 모드로 로그인합니다. 클린 프롬프트를 가져오려면 **Enter** 를 눌러야 할 수 있습니다.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@workstation ~]#
```

▶ 6. 복구 모드에서는 루트 파일 시스템이 읽기/쓰기 모드에 있습니다.

```
[root@workstation ~]# mount
...output omitted...
/dev/vda4 on / type xfs
(rw,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
...output omitted...
```

▶ 7. **Ctrl+d**를 눌러 부팅 프로세스를 계속합니다.

그래픽 로그인 화면이 표시됩니다. **student** 사용자로 로그인합니다.

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish boot-selecting
```

이것으로 섹션을 완료합니다.

루트 암호 재설정

목표

현재 루트 암호가 분실된 경우 시스템에 로그인하여 루트 암호를 변경합니다.

부트 로더에서 루트 암호 재설정

모든 시스템 관리자가 할 수 있어야 하는 작업 중 하나는 분실한 **root** 암호를 재설정하는 것입니다. 관리자가 권한이 없지만 **sudo**에 대한 전체 액세스 권한이 있는 사용자로 로그인되어 있거나 이미 **root**로 로그인되어 있는 경우 이 작업은 간단합니다. 관리자가 로그인되어 있지 않은 경우는 작업이 약간 더 복잡합니다.

새 **root** 암호를 설정하는 방법은 여러 가지가 있습니다. 시스템 관리자는 예를 들면 라이브 CD를 사용하여 시스템을 부팅한 다음 여기에서 루트 파일 시스템을 마운트하고 **/etc/shadow**를 편집할 수 있습니다. 이 섹션에서는 외부 미디어가 필요하지 않은 방법을 살펴봅니다.

Red Hat Enterprise Linux 9에서는 **initramfs** 이미지에서 실행되는 스크립트가 특정 지점에서 일시 중지되고 **root** 쉘을 제공한 다음 해당 쉘이 종료될 때 계속되도록 할 수 있습니다. 이 스크립트는 대부분 디버깅을 위한 것이지만 잊어버린 **root** 암호를 재설정하는 데 이 방법을 사용할 수도 있습니다.

Red Hat Enterprise Linux 9부터는 DVD에서 시스템을 설치하는 경우 유지 관리 모드로 전환하려고 할 때 기본 커널에서 **root** 암호를 요청합니다. 따라서 분실한 **root** 암호를 재설정하려면 복구 커널을 사용해야 합니다.

해당 **root** 쉘에 액세스하려면 다음 단계를 수행합니다.

1. 시스템을 재부팅합니다.
2. **Enter** 키를 제외한 임의의 키를 눌러서 부트 로더 카운트다운을 중단합니다.
3. 커서를 복구 커널 항목(이름에 **rescue**라는 단어가 있는 항목)으로 이동하여 부팅합니다.
4. **e** 를 눌러 선택한 항목을 편집합니다.
5. 커서를 커널 명령줄(**linux**로 시작하는 행)로 이동합니다.
6. **rd.break**을 추가합니다. 이 옵션을 사용하면 **initramfs** 이미지에서 실제 시스템으로 제어 권한이 이동하기 직전에 중단됩니다.
7. **Ctrl+x** 를 눌러 변경 사항을 적용하여 부팅합니다.
8. 메시지가 표시되면 **Enter** 키를 눌러 유지 관리를 수행합니다.

이때 시스템에 **root** 쉘이 있고 디스크의 루트 파일 시스템은 **/sysroot**에 읽기 전용으로 마운트됩니다. 트러블슈팅에서는 종종 루트 파일 시스템을 수정해야 하므로 루트 파일 시스템을 읽기/쓰기로 다시 마운트해야 합니다. 다음 단계에서는 **mount** 명령에 대한 **remount**, **rw** 옵션이 새 옵션(**rw**)을 사용하여 파일 시스템을 다시 마운트하는 방법을 보여줍니다.

**중요**

시스템이 SELinux를 아직 활성화하지 않았으므로 생성한 어떤 파일에도 SELinux 컨텍스트가 없습니다. 일부 툴(예: **passwd** 명령)은 먼저 임시 파일을 생성한 다음 편집하려는 파일로 교체하여 SELinux 컨텍스트 없이 파일을 효과적으로 생성합니다. 따라서 **passwd** 명령을 **rd.break**와 함께 사용하면 **/etc/shadow** 파일에 SELinux 컨텍스트가 수신되지 않습니다.

root 암호를 재설정하려면 다음 절차를 따릅니다.

1. **/sysroot** 를 읽기/쓰기로 다시 마운트합니다.

```
sh-5.1# mount -o remount,rw /sysroot
```

2. **/sysroot**가 파일 시스템 트리의 루트로 간주되는 **chroot** 환경으로 전환합니다.

```
sh-5.1# chroot /sysroot
```

3. 새 **root** 암호를 설정합니다.

```
sh-5.1# passwd root
```

4. 레이블이 지정되지 않은 모든 파일(이 시점에서는 **/etc/shadow** 포함)에 부팅 중 다시 레이블이 지정되도록 합니다.

```
sh-5.1# touch /.autorelabel
```

5. **exit** 을 두 번 입력합니다. 첫 번째는 **chroot** 환경을 종료하고 두 번째는 **initramfs** 디버그 쉘을 종료합니다.

이때 시스템은 부팅을 계속하면서 전체 SELinux 레이블을 재지정한 다음 다시 재부팅합니다.

클라우드 이미지 기반 시스템 복구

설치 프로그램을 사용하는 대신 공식 클라우드 이미지 중 하나를 배포하고 수정하여 시스템을 설치한 경우 부팅 프로세스에서 일부 시스템 구성 측면이 다를 수 있습니다.

rd.break 옵션을 사용하여 루트 쉘을 가져오는 절차는 이전에 설명한 절차와 유사하지만 약간의 변경 사항이 있습니다.

시스템이 Red Hat Enterprise Linux 클라우드 이미지에서 배포된 경우 부팅 메뉴에 기본적으로 복구 커널이 없습니다. 그러나 기본 커널을 사용하여 root 암호를 입력하지 않고 **rd.break** 옵션을 사용하여 유지 관리 모드로 전환할 수 있습니다.

커널이 부팅 메시지를 인쇄하고 시스템 콘솔에 root 프롬프트를 표시합니다. 사전 빌드된 이미지에는 부트로더의 커널 명령줄에 여러 개의 **console=** 인수가 있을 수 있습니다. 시스템에서 커널 메시지를 모든 콘솔에 전송하더라도 **rd.break** 옵션을 설정하는 루트 쉘은 명령줄에 지정된 마지막 콘솔을 사용합니다. 프롬프트가 표시되지 않으면 부트 로더에서 커널 명령줄을 편집할 때 일시적으로 **console=** 인수를 재정렬할 수 있습니다.

로그 검사

이전에 실패한 부팅의 로그를 확인하는 것이 유용할 수 있습니다. 재부팅 후에도 시스템 저널이 영속적인 경우 `journalctl` 도구를 사용하여 해당 로그를 검사할 수 있습니다.

기본적으로 시스템 저널은 `/run/log/journal` 디렉터리에 보관되므로 시스템이 재부팅되면 저널이 삭제됩니다. 부팅 후에도 지속되는 `/var/log/journal` 디렉터리에 저널을 저장하려면 `/etc/systemd/journald.conf` 파일에서 `Storage` 매개 변수를 `persistent`로 설정하십시오.

```
[root@host ~]# vim /etc/systemd/journald.conf
...
[Journal]
Storage=persistent
...
[root@host ~]# systemctl restart systemd-journald.service
```

이전 부팅의 로그를 검사하려면 `journalctl` 명령에 `-b` 옵션을 사용합니다. 인수 없이 `journalctl` 명령에 `-b` 옵션을 사용하면 마지막 부팅 이후의 메시지만 표시됩니다. 인수로 음수를 사용하면 이전 부팅의 로그가 표시됩니다.

```
[root@host ~]# journalctl -b -1 -p err
```

이 명령은 이전 부팅에서 오류 또는 그 이상의 심각한 문제로 평가된 모든 메시지를 표시합니다.

Systemd 부팅 문제 해결

Red Hat Enterprise Linux 8 이후 버전에서는 다음 툴을 제공하므로 부팅 시 서비스 시작 문제를 해결할 수 있습니다.

초기 디버그 쉘 활성화

`debug-shell` 서비스를 `systemctl enable debug-shell.service` 명령과 함께 활성화하면 부팅 시퀀스 중 시작 단계에서 TTY9 (`Ctrl+Alt+F9`)에 `root` 쉘이 생성됩니다. 운영 체제가 부팅되는 동안 관리자가 시스템을 디버깅할 수 있도록 이 쉘은 자동으로 `root`로 로그인됩니다.



경고

디버깅을 마치면 `debug-shell.service` 서비스를 비활성화합니다. 그러지 않으면 인증되지 않은 `root` 쉘이 로컬 콘솔 액세스 권한이 있는 모든 사용자에게 열린 상태로 유지됩니다.

GRUB2 메뉴를 사용하여 부팅하는 동안 디버그 쉘을 활성화하려면 다음 단계를 따르십시오.

1. 시스템을 재부팅합니다.
2. **Enter** 키를 제외한 임의의 키를 눌러서 부트 로더 카운트다운을 중단합니다.
3. 커서를 부팅할 커널 항목으로 이동합니다.
4. **e** 를 눌러 선택한 항목을 편집합니다.
5. 커서를 커널 명령줄(`linux`로 시작하는 행)로 이동합니다.
6. `systemd.debug-shell`을 추가합니다. 이 매개 변수를 사용하면 시스템이 디버그 쉘로 부팅됩니다.

7. **Ctrl+x** 를 눌러 변경 사항을 적용하여 부팅합니다.

긴급 및 복구 타겟 사용

부트 로더에서 커널 명령줄에 **systemd.unit=rescue.target** 또는 **systemd.unit=emergency.target** 을 추가하면 시스템이 정상적으로 시작하지 않고 복구 또는 긴급 헬로 들어갑니다. 두 헬 모두 **root** 암호가 필요합니다.

긴급 타겟은 루트 파일 시스템을 읽기 전용으로 마운트하지만 복구 타겟은 더 많은 시스템(예: 로깅 서비스, 파일 시스템)이 초기화되도록 **sysinit.target** 유닛이 완료될 때까지 기다립니다. 이 시점의 루트 사용자는 **mount -o remount,rw /** 명령을 사용하여 드라이브를 읽기 쓰기 상태로 다시 마운트할 때까지 **/etc/fstab** 을 변경할 수 없습니다.

관리자는 이러한 헬을 사용하여 시스템이 정상적으로 부팅되지 않는 문제를 해결할 수 있습니다. 이러한 문제의 예로는 서비스 간 종속성 반복문 또는 **/etc/fstab**의 잘못된 입력 항목이 있습니다. 이러한 헬에서 종료 할 경우 일반 부팅 프로세스로 계속됩니다.

중단된 작업 확인

시작하는 동안 **systemd** 에서 여러 작업이 생성됩니다. 이러한 작업 중 일부가 완료되지 않으면 이로 인해 다른 작업이 실행되지 않습니다. 관리자는 **systemctl list-jobs** 명령을 사용하여 현재 작업 목록을 검사할 수 있습니다. **running**으로 나열된 작업은 **waiting**으로 나열된 작업을 계속하기 전에 완료해야 합니다.



참조

[dracut cmdline\(7\)](#), [systemd-journald\(8\)](#), [journald.conf\(5\)](#),
[journalctl\(1\)](#), [systemctl\(1\)](#) 도움말 페이지

▶ 연습 가이드

루트 암호 재설정

이 연습에서는 시스템에 **root** 암호를 재설정합니다.

결과

- 분실한 **root** 사용자 암호를 재설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 네트워크의 **servera** 시스템에 연결할 수 있는지 확인하는 시작 스크립트를 실행합니다. 또한 **root** 암호를 임의의 문자열로 재설정하고 GRUB2 메뉴의 시간 제한을 더 높게 설정합니다.

```
[student@workstation ~]$ lab start boot-resetting
```

지침

▶ 1. **servera**를 재부팅하고 부트 로더 메뉴에서 카운트다운을 중단합니다.

1. 강의실 환경에 적합한 **servera** 콘솔의 아이콘을 찾아 콘솔을 엽니다.

관련 버튼 또는 메뉴 항목을 사용하여 시스템에 **Ctrl+Alt+Del**을 전송합니다.

2. 부트 로더 메뉴가 나타나면 **Enter** 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.

▶ 2. 커널에서 모든 파일 시스템을 마운트한 직후 제어 권한을 **systemd**로 전달하기 전에 메모리의 복구 커널 부트 로더 항목을 편집하여 부트 프로세스를 중단합니다.

1. 커서 키를 사용하여 복구 커널 항목(이름에 **rescue**라는 단어가 있는 항목)을 강조 표시합니다.

2. **e**를 눌러 현재 항목을 편집합니다.

3. 커서 키를 사용하여 **linux**로 시작하는 행으로 이동합니다.

4. **End**를 눌러 커서를 행 끝으로 이동합니다.

5. 행 끝에 **rd.break**을 추가합니다.



참고

콘솔에서 텍스트를 보기 어려운 경우 부트 로더 항목에서 커널 행을 편집할 때 해상도를 변경하는 것이 좋습니다.

콘솔 해상도를 변경하려면 **linux** 단어로 시작하는 행에서 **rd.break** 다음에 **video=640x480** 또는 **vga=ask**를 추가합니다. 대부분의 콘솔에서는 **640x480** 해상도로 충분합니다. **vga=ask**를 사용하여 환경에 더 적합한 해상도를 선택할 수 있습니다.

2.6. **Ctrl+x** 를 눌러 수정된 구성으로 부팅합니다.

- ▶ 3. **Enter** 를 눌러 유지 관리를 수행합니다. **sh-5.1#** 프롬프트에서 **/sysroot** 파일 시스템을 읽기/쓰기로 다시 마운트한 다음 **chroot** 명령을 사용하여 **/sysroot**의 **chroot** 환경으로 이동합니다.

```
sh-5.1# mount -o remount,rw /sysroot
...output omitted...
sh-5.1# chroot /sysroot
```

- ▶ 4. **root** 암호를 다시 **redhat**으로 변경합니다.

```
sh-5.1# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- ▶ 5. 부팅 후 시스템에서 전체 SELinux 레이블을 자동으로 다시 지정하도록 구성합니다. 이 단계는 **passwd** 명령이 SELinux 컨텍스트 없이 **/etc/shadow** 파일을 다시 만들기 때문에 필요합니다.

```
sh-5.1# touch /.autorelabel
```

- ▶ 6. **exit** 을 두 번 입력해서 시스템을 계속해서 정상적으로 부팅합니다. 시스템에서 SELinux 레이블 재지정 작업을 실행한 다음 자동으로 재부팅합니다. 시스템이 작동되면 콘솔에서 **root** 로 로그인하여 작업을 확인합니다.

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish boot-resetting
```

이것으로 섹션을 완료합니다.

부팅 시 파일 시스템 문제 복구

목표

부팅 프로세스를 중단하는 파일 시스템 구성 또는 손상 문제를 수동으로 복구합니다.

파일 시스템 문제

부팅 프로세스 중 **systemd** 서비스는 **/etc/fstab** 파일에 정의된 영구 파일 시스템을 마운트합니다.

/etc/fstab 파일의 오류 또는 손상된 파일 시스템은 시스템을 차단하여 부팅 프로세스가 완료되지 않도록 할 수 있습니다. 일부 오류 시나리오에서는 시스템이 부팅 프로세스를 중단하고 **root** 사용자 암호가 필요한 긴급 쉘을 엽니다.

다음 목록에서는 부팅 프로세스 중 **/etc/fstab** 파일을 구문 분석할 때 발생하는 몇 가지 일반적인 파일 시스템 마운트 문제를 설명합니다.

손상된 파일 시스템

systemd 서비스에서 파일 시스템 복구를 시도합니다. 문제를 자동으로 복구할 수 없는 경우 시스템에서 비상 쉘을 엽니다.

장치 또는 UUID가 존재하지 않음

장치를 사용할 수 있을 때까지 기다리는 동안 **systemd** 서비스가 시간 초과됩니다. 장치가 응답하지 않으면 비상 쉘이 열립니다.



참고

마운트 지점이 없는 경우 Red Hat Enterprise Linux 9는 부팅 프로세스 중 마운트 지점을 자동으로 생성합니다.

부팅 시 파일 시스템 문제 복구

systemd 아키텍처에서는 파일 시스템 문제로 인해 부팅을 완료할 수 없는 시스템에 액세스하기 위해 액세스에 **root** 암호가 필요한 긴급 쉘을 여는 **emergency** 부팅 타겟을 제공합니다.

다음 예제에서는 시스템이 파일 시스템 문제를 발견하고 **emergency** 타겟으로 전환할 때의 부팅 프로세스 출력을 보여줍니다.

```
...output omitted...
[*      ] A start job is running for /dev/vda2 (27s / 1min 30s)
[ TIME ] Timed out waiting for device /dev/vda2.
[DEPEND] Dependency failed for /mnt/mountfolder
[DEPEND] Dependency failed for Local File Systems.
[DEPEND] Dependency failed for Mark need to relabel after reboot.
...output omitted...
[ OK   ] Started Emergency Shell.
[ OK   ] Reached target Emergency Mode.
...output omitted...
Give root password for maintenance
(or press Control-D to continue):
```

systemd 데몬이 `/dev/vda2` 장치를 마운트하는 데 실패했으며 시간이 초과되었습니다. 장치를 사용할 수 없기 때문에 시스템에서 유지 관리 액세스를 위해 비상 쉘을 엽니다.

시스템에서 긴급 쉘을 열 때 파일 시스템 문제를 복구하려면 먼저 잘못된 파일 시스템을 찾아 결함을 확인하고 복구하십시오. 이제 **systemd** 구성을 다시 로드하여 자동 마운트를 다시 시도합니다.

mount 명령을 사용하여 **systemd** 데몬에서 현재 마운트한 파일 시스템을 찾습니다.

```
[root@host ~]# mount
...output omitted...
/dev/vda1 on / type xfs (ro,relatime,seclabel,attr2,inode64,noquota)
...output omitted...
```

루트 파일 시스템이 `ro` (읽기 전용) 옵션을 사용하여 마운트된 경우 `/etc/fstab` 파일을 편집할 수 없습니다. 필요한 경우 `/etc/fstab` 파일을 열기 전에 `rw` (읽기/쓰기) 옵션을 사용하여 루트 파일 시스템을 임시로 다시 마운트합니다. `remount` 옵션을 사용하면 사용 중인 파일 시스템에서 파일 시스템을 마운트 해제하지 않고도 마운트 매개 변수를 변경할 수 있습니다.

```
[root@host ~]# mount -o remount,rw /
```

mount --all 옵션을 사용하여 `/etc/fstab` 파일에 나열된 모든 파일 시스템을 마운트합니다. 이 옵션은 모든 파일 시스템 항목에 프로세스를 마운트하지만 이미 마운트된 항목은 건너뜁니다. 이 명령은 파일 시스템을 마운트할 때 발생하는 오류를 표시합니다.

```
[root@host ~]# mount --all
mount: /mnt/mountfolder: mount point does not exist.
```

이 시나리오에서는 `/mnt/mountfolder` 마운트 디렉터리가 없습니다. 마운트를 다시 시도하기 전에 `/mnt/mountfolder` 디렉터리를 생성하십시오. 항목의 오타 또는 잘못된 장치 이름 또는 UUID를 포함하여 기타 오류 메시지가 표시될 수 있습니다.

`/etc/fstab` 파일의 모든 문제를 수정한 후 `systemctl daemon-reload` 명령을 사용하여 새 `/etc/fstab` 파일을 등록하도록 **systemd** 데몬에 알립니다. 그런 다음 모든 항목을 다시 마운트합니다.

```
[root@host ~]# systemctl daemon-reload
[root@host ~]# mount --all
```



참고

systemd 서비스는 각 항목을 `.mount` 유형의 **systemd** 유닛 구성으로 변환한 다음 유닛을 서비스로 시작하여 `/etc/fstab` 파일을 처리합니다. `daemon-reload` 옵션은 **systemd** 데몬에 모든 유닛 구성을 다시 빌드하여 로드하도록 요청합니다.

mount --all 명령이 추가 오류 없이 성공하는 경우 마지막 테스트는 시스템 부팅 중 파일 시스템 마운트가 성공적으로 수행되었는지 확인하는 것입니다. 시스템을 재부팅하고 부팅이 정상적으로 완료될 때까지 기다립니다.

```
[root@host ~]# systemctl reboot
```

`/etc/fstab` 파일에서 빠른 테스트를 수행하려면 `nofail` 마운트 항목 옵션을 사용하십시오. `/etc/fstab` 항목에 `nofail` 옵션을 사용하면 해당 파일 시스템이 성공적으로 마운트되지 않아도 시스템이 부팅됩니다. 이 옵션은 항상 마운트해야 하는 프로덕션 파일 시스템과 함께 사용해서는 안 됩니다. `nofail` 옵션을 사용하면 애플리케이션이 파일 시스템 데이터가 누락된 상태로 시작되어 심각한 결과가 발생할 수 있습니다.



참조

`systemd-fsck(8)`, `systemd-fstab-generator(8)` 및 `systemd.mount(5)` 도움말
페이지

▶ 연습 가이드

부팅 시 파일 시스템 문제 복구

이 연습에서는 부팅 프로세스가 실패한 `/etc/fstab`의 잘못된 구성에서 시스템을 복구합니다.

결과

- `/etc/fstab` 파일 문제를 진단하고 긴급 모드를 사용하여 시스템을 복구합니다.

시작하기 전에

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start boot-repairing
```

지침

▶ 1. `servera` 시스템 콘솔에 액세스하여 부팅 프로세스가 초기에 중단되는지 확인합니다.

1. 강의실 환경에 해당하는 `servera` 콘솔의 아이콘을 찾습니다. 콘솔을 엽니다.
시작 작업이 완료되지 않은 것을 확인합니다. 이 동작의 원인이 무엇인지 생각합니다.
2. 관련 버튼 또는 메뉴 항목을 사용하여 시스템에 `Ctrl+Alt+Del`을 보내 `servera` 시스템을 재부팅합니다. 이러한 부팅 문제의 경우 주요 시퀀스에서 실행 중인 작업을 즉시 중단하지 않을 수도 있으며, 시스템을 재부팅하려면 시간 초과되기를 기다려야 할 수 있습니다.
`Ctrl+Alt+Del`을 전송하지 않고 작업이 시간 초과될 때까지 기다리면 시스템에서 자동으로 긴급 쉘이 생성됩니다.
3. 부트 로더 메뉴가 나타나면 `Enter` 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.

▶ 2. 이전 부팅에서 발생한 오류를 보면 시스템 일부는 아직 작동하고 있습니다. `redhat`을 `root` 사용자 암호로 사용하여 긴급 부팅을 시도합니다.

1. 커서 키를 사용하여 기본 부트 로더 항목을 강조 표시합니다.
2. `e` 키를 눌러 현재 항목을 편집합니다.
3. 커서 키를 사용하여 `linux`라는 단어로 시작하는 행으로 이동합니다.
4. `End`를 눌러 커서를 행 끝으로 이동합니다.
5. 행 끝에 `systemd.unit=emergency.target` 문자열을 추가합니다.

**참고**

콘솔에서 텍스트를 보기 어려운 경우 부트 로더 항목에서 커널 행을 편집할 때 해상도를 변경하는 것이 좋습니다.

콘솔 해상도를 변경하려면 **linux** 단어로 시작하는 행에서 **systemd.unit=emergency.target** 다음에 **video=640x480** 또는 **vga=ask** 를 추가합니다. 대부분의 콘솔에서는 **640x480** 해상도로 충분합니다. **vga=ask**를 사용하여 환경에 더 적합한 해상도를 선택할 수 있습니다.

2.6. **Ctrl+x** 를 눌러 수정된 구성으로 부팅합니다.

▶ 3. 긴급 모드로 로그인합니다.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@servera ~]#
```

▶ 4. **systemd** 데몬에서 현재 마운트하는 파일 시스템을 확인합니다. **systemd** 데몬은 루트 파일 시스템을 읽기 전용 모드로 마운트합니다.

```
[root@servera ~]# mount
...output omitted...
/dev/vda4 on / type xfs
(ro,relatime,seclabel,attr2,inode64,logbufs=8,logbsize=32k,noquota)
...output omitted...
```

▶ 5. 루트 파일 시스템을 읽기/쓰기 모드로 다시 마운트합니다.

```
[root@servera ~]# mount -o remount,rw /
```

▶ 6. 나머지 모든 파일 시스템을 마운트해 봅니다. **--all (-a)** 옵션은 **/etc/fstab** 파일에 나열되어 있고 아직 마운트되지 않은 모든 파일 시스템을 마운트합니다.

```
[root@servera ~]# mount -a
mount: /RemoveMe: mount point does not exist.
```

▶ 7. **/etc/fstab** 파일을 편집하여 문제를 해결합니다.

7.1. **vim /etc/fstab** 명령을 사용하여 잘못된 줄을 제거하거나 주석 처리합니다.

```
[root@servera ~]# cat /etc/fstab
...output omitted...
# /dev/sdz1    /RemoveMe    xfs    defaults    0 0
```

7.2. 시스템의 **systemd** 데몬을 다시 로드하여 새 **/etc/fstab** 파일 구성을 등록합니다.

```
[root@servera ~]# systemctl daemon-reload
```

- ▶ 8. 모든 항목의 마운트를 시도하여 이제 **/etc/fstab**이 올바른지 확인합니다.

```
[root@servera ~]# mount -a
```

- ▶ 9. 시스템을 재부팅하고 부팅이 완료될 때까지 기다립니다. 이제 시스템이 정상적으로 부팅됩니다.

```
[root@servera ~]# systemctl reboot
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish boot-repairing
```

이것으로 섹션을 완료합니다.

▶ 랩

부팅 프로세스 제어

이 랩에서는 시스템의 **root** 암호를 재설정하고 잘못된 구성에서 복구한 다음 기본 부팅 타겟을 설정합니다.

결과

- **root** 사용자가 분실한 암호를 재설정합니다.
- 부팅 문제를 진단하고 해결합니다.
- 기본 **systemd** 타겟을 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start boot-review
```

지침

1. **serverb** 시스템에서 **root** 사용자의 암호를 **redhat** 으로 재설정합니다.
강의실 환경에 적합한 **serverb** 시스템 콘솔의 아이콘을 찾아 콘솔을 엽니다.
2. Boot-loader(부트 로더) 메뉴에서 Default Kernel Boot-loader(기본 커널 부트 로더) 항목을 선택합니다. 시작 작업이 성공적으로 완료되지 않아 시스템이 부팅되지 않습니다. **serverb** 시스템의 콘솔에서 문제를 해결합니다.
3. 해당 시스템에서 부팅 시 그래픽 인터페이스가 자동으로 시작되도록 **serverb** 시스템의 기본 **systemd** 타겟을 변경합니다.
serverb 시스템에 그래픽 인터페이스가 설치되어 있지 않습니다. 기본 타겟만 설정하고 패키지는 설치하지 않습니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade boot-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish boot-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

부팅 프로세스 제어

이 랩에서는 시스템의 **root** 암호를 재설정하고 잘못된 구성에서 복구한 다음 기본 부팅 타겟을 설정합니다.

결과

- **root** 사용자가 분실한 암호를 재설정합니다.
- 부팅 문제를 진단하고 해결합니다.
- 기본 **systemd** 타겟을 설정합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start boot-review
```

지침

1. serverb 시스템에서 root 사용자의 암호를 redhat 으로 재설정합니다.

강의실 환경에 적합한 **serverb** 시스템 콘솔의 아이콘을 찾아 콘솔을 엽니다.

1. 관련 버튼 또는 메뉴 항목을 사용하여 시스템에 **Ctrl+Alt+Del** 을 전송합니다.
2. 부트 로더 메뉴가 나타나면 **Enter** 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.
3. 커서 키를 사용하여 복구 커널 부트 로더 항목(이름에 rescue 라는 단어가 있는 항목)을 강조 표시합니다.
4. **e** 를 눌러 현재 항목을 편집합니다.
5. 커서 키를 사용하여 **linux**라는 텍스트로 시작하는 행으로 이동합니다.
6. **Ctrl+e**를 눌러 커서를 행 끝으로 이동합니다.
7. 행 끝에 **rd.break** 이라는 텍스트를 추가합니다.



참고

콘솔에서 텍스트를 보기 어려운 경우 부트 로더 항목에서 커널 행을 편집할 때 해상도를 변경하는 것이 좋습니다.

콘솔 해상도를 변경하려면 **linux** 단어로 시작하는 행에서 **rd.break** 다음에 **video=640x480** 또는 **vga=ask** 를 추가합니다. 대부분의 콘솔에서는 **640x480** 해상도로 충분합니다. **vga=ask**를 사용하여 환경에 더 적합한 해상도를 선택할 수 있습니다.

- 1.8. **Ctrl+x**를 눌러 수정된 구성으로 부팅합니다.
- 1.9. **Enter** 키를 눌러 유지 관리 모드로 전환합니다.
- 1.10. **sh-5.1** 프롬프트에서 **/sysroot** 파일 시스템을 쓰기 가능으로 다시 마운트한 다음 **/sysroot** 디렉터리에 **chroot** 명령을 사용합니다.

```
sh-5.1# mount -o remount,rw /sysroot
...output omitted...
sh-5.1# chroot /sysroot
```

- 1.11. **root** 사용자의 암호를 **redhat** 으로 설정합니다.

```
sh-5.1# passwd root
Changing password for user root.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 1.12. 부팅 후 시스템에서 전체 SELinux 레이블을 다시 지정하도록 구성합니다.

```
sh-5.1# touch /.autorelabel
```

- 1.13. **chroot** 환경 및 **sh-5.1** 프롬프트를 종료합니다. 파일 시스템의 레이블을 다시 지정하면 시스템에서 유지 관리 모드로 전환하라는 메시지를 표시합니다. 그러나 기다리면 재부팅이 완료되고 boot-loader 메뉴가 표시됩니다.
- 2.** Boot-loader(부트 로더) 메뉴에서 Default Kernel Boot-loader(기본 커널 부트 로더) 항목을 선택합니다. 시작 작업이 성공적으로 완료되지 않아 시스템이 부팅되지 않습니다. **serverb** 시스템의 콘솔에서 문제를 해결합니다.

- 2.1. 시스템을 긴급 모드로 부팅합니다. 관련 버튼 또는 메뉴 항목을 사용하여 시스템에 **Ctrl+Alt+Del** 을 보내 **serverb** 시스템을 재부팅합니다.
- 2.2. 부트 로더 메뉴가 나타나면 **Enter** 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.
- 2.3. 커서 키를 사용하여 기본 부트 로더 항목을 강조 표시합니다.
- 2.4. **e** 를 눌러 현재 항목을 편집합니다.
- 2.5. 커서 키를 사용하여 **linux**라는 텍스트로 시작하는 행으로 이동합니다.
- 2.6. **Ctrl+e**를 눌러 커서를 행 끝으로 이동합니다.
- 2.7. 행 끝에 **systemd.unit=emergency.target** 이라는 텍스트를 추가합니다.
- 2.8. **Ctrl+x**를 눌러 수정된 구성으로 부팅합니다.
- 2.9. 긴급 모드로 로그인합니다.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

2.10. / 파일 시스템을 쓰기 가능으로 다시 마운트합니다.

```
[root@serverb ~]# mount -o remount,rw /
...output omitted...
```

2.11. 모든 파일 시스템을 마운트합니다.

```
[root@serverb ~]# mount -a
mount: /olddata: can't find UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc.
```

2.12. /etc/fstab 파일을 편집하여 /olddata 마운트 지점을 마운트하는 잘못된 행을 제거하거나 주석 처리합니다.

```
[root@serverb ~]# vim /etc/fstab
...output omitted...
#UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc  /olddata  xfs  defaults  0 0
```

2.13. 시스템의 **systemd** 데몬을 업데이트하여 /etc/fstab 파일 구성에 변경 사항을 등록합니다.

```
[root@serverb ~]# systemctl daemon-reload
```

2.14. 모든 항목의 마운트를 시도하여 /etc/fstab 파일 구성이 올바른지 확인합니다.

```
[root@serverb ~]# mount -a
```

2.15. 시스템을 재부팅하고 부팅이 완료될 때까지 기다립니다. 이제 시스템이 정상적으로 부팅됩니다.

```
[root@serverb ~]# systemctl reboot
```

3. 해당 시스템에서 부팅 시 그래픽 인터페이스가 자동으로 시작되도록 **serverb** 시스템의 기본 **systemd** 타겟을 변경합니다.

serverb 시스템에 그래픽 인터페이스가 설치되어 있지 않습니다. 기본 타겟만 설정하고 패키지는 설치하지 않습니다.

3.1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

3.2. **graphical.target**을 기본 타겟으로 설정합니다.

```
[root@serverb ~]# systemctl set-default graphical.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/
graphical.target.
```

3.3. 올바른 기본값이 설정되었는지 확인합니다.

```
[root@serverb ~]# systemctl get-default
graphical.target
```

3.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade boot-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish boot-review
```

이것으로 섹션을 완료합니다.

요약

- **systemd** 유ти리티는 부팅할 때 및 실행 중인 시스템에서 시스템 리소스, 서버 데몬 및 기타 프로세스를 활성화하는 방법을 제공합니다.
- **systemctl** 유ти리티를 사용하여 서비스를 시작, 중지, 다시 로드, 활성화 및 비활성화합니다.
- **systemd** 유ти리티를 사용하여 서비스 유닛, 소켓 유닛 및 경로 유닛을 관리합니다.
- **systemctl status** 명령을 사용하여 **systemd** 유ти리티에서 시작된 시스템 데몬 및 네트워크 서비스의 상태를 확인합니다.
- **systemctl list-dependencies** 명령은 특정 서비스 유닛이 종속된 모든 서비스 유닛을 표시합니다.
- **systemd** 유ти리티는 종속성을 충족해야 하는 경우에도 실행되지 않도록 서비스 유닛을 마스킹할 수 있습니다.
- **systemctl reboot** 및 **systemctl poweroff** 명령은 각각 시스템을 재부팅하고 전원을 끕니다.
- **systemctl isolate target-name.target** 명령은 런타임 시 새 타겟으로 전환합니다.
- **systemctl get-default** 및 **systemctl set-default** 명령으로 기본 타겟을 쿼리하고 설정할 수 있습니다.
- 커널 명령줄에 **rd.break** 옵션을 사용하여 **initramfs** 이미지에서 제어가 넘어오기 전에 부팅 프로세스를 중단할 수 있습니다. 루트 파일 시스템이 **/sysroot** 디렉터리에 읽기 전용으로 마운트됩니다.
- 긴급 타겟을 사용하여 파일 시스템 문제를 진단 및 해결할 수 있습니다.

로그 분석 및 저장

목적

문제 해결을 위해 시스템 이벤트 로그를 찾아 정확히 해석합니다.

목표

- 이벤트를 기록할 Red Hat Enterprise Linux 기본 로깅 아키텍처에 대해 설명합니다.
- 문제를 해결하거나 시스템 상태를 검토하기 위해 관련 syslog 파일의 이벤트를 해석합니다.
- 문제를 해결하거나 시스템 상태를 검토하기 위해 시스템 저널의 로그 항목을 찾아 해석합니다.
- 서버가 재부팅될 때 이벤트 레코드를 보존하도록 시스템 저널을 구성합니다.
- NTP(Network Time Protocol)를 사용하여 정확한 시간 동기화를 유지하고 시스템 저널 및 로그에서 기록하는 이벤트의 타임스탬프가 정확하도록 시간대를 구성합니다.

섹션

- 시스템 로그 아키텍처 설명(퀴즈)
- syslog 파일 검토(안내에 따른 연습)
- 시스템 저널 항목 검토(안내에 따른 연습)
- 시스템 저널 보존(안내에 따른 연습)
- 정확한 시간 유지 관리(안내에 따른 연습)

랩

로그 분석 및 저장

시스템 로그 아키텍처 설명

목표

이벤트를 기록할 Red Hat Enterprise Linux 기본 로깅 아키텍처에 대해 설명합니다.

시스템 로깅

운영 체제 커널 및 기타 프로세스는 시스템이 실행 중일 때 발생하는 이벤트의 로그를 기록합니다. 이러한 로그는 시스템 감사 및 문제 해결에 사용됩니다. **less** 및 **tail** 명령과 같은 텍스트 유ти리티를 사용하여 이러한 로그를 검사할 수 있습니다.

Red Hat Enterprise Linux는 syslog 프로토콜을 기반으로 하는 표준 로깅 시스템을 사용하여 시스템 메시지를 기록합니다. 많은 프로그램이 로깅 시스템을 사용하여 이벤트를 기록하고 로그 파일로 구성합니다.

systemd-journald 및 **rsyslog** 서비스는 Red Hat Enterprise Linux 9에서 syslog 메시지를 처리합니다.

systemd-journald 서비스는 운영 체제 이벤트 로깅 아키텍처의 핵심입니다. **systemd-journald** 서비스는 여러 소스에서 이벤트 메시지를 수집합니다.

- 시스템 커널
- 부팅 프로세스 초기 단계의 출력
- 데몬의 표준 출력 및 표준 오류
- syslog 이벤트

systemd-journald 서비스는 로그를 표준 형식으로 재구성하고 인덱싱된 구조적 시스템 저널에 씁니다. 기본적으로 이 저널은 재부팅 후 유지되지 않는 파일 시스템에 저장됩니다.

rsyslog 서비스는 **systemd-journald** 서비스가 저널에서 받은 syslog 메시지를 도착하는 대로 읽습니다. 그런 다음 **rsyslog** 서비스는 syslog 이벤트를 처리하여 로그 파일에 기록하거나 자체 구성에 따라 다른 서비스에 전달합니다.

rsyslog 서비스는 재부팅 후에도 지속되는 **/var/log** 디렉터리의 로그 파일에 syslog 메시지를 정렬하고 씁니다. 또한 이 서비스는 각 메시지를 전송한 프로그램 유형 및 각 syslog 메시지의 우선 순위에 따라 로그 메시지를 특정 로그 파일에 정렬합니다.

syslog 메시지 파일 외에도 **/var/log** 디렉터리에는 시스템의 다른 서비스의 로그 파일이 포함되어 있습니다. 다음 테이블은 **/var/log** 디렉터리에 유용한 몇 개의 파일을 나열합니다.

선택한 시스템 로그 파일

로그 파일	저장되는 메시지 유형
/var/log/messages	대부분의 syslog 메시지가 여기에 기록됩니다. 단, 인증 및 이메일 처리와 예약된 작업 실행에 대한 메시지, 순수하게 디버깅에만 관련된 메시지는 예외입니다.
/var/log/secure	보안 및 인증 이벤트에 대한 syslog 메시지
/var/log/maillog	메일 서버에 대한 syslog 메시지

로그 파일	저장되는 메시지 유형
/var/log/cron	예약된 작업 실행에 대한 syslog 메시지
/var/log/boot.log	시스템 시작에 대한 비 syslog 콘솔 메시지

일부 애플리케이션은 **syslog** 서비스를 사용하여 로그 메시지를 관리하지 않습니다. 예를 들어 Apache 웹 서버는 **/var/log** 디렉터리의 하위 디렉터리에 있는 파일에 로그 메시지를 저장합니다.



참조

systemd-journald.service(8), **rsyslogd(8)** 및 **rsyslog.conf(5)** 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index

에서 Red Hat Enterprise Linux 9 Configuring Basic System Settings 가이드의 Troubleshooting Problems Using Log Files 섹션을 참조하십시오.

▶ 퀴즈

시스템 로그 아키텍처 설명

다음 질문에 대해 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 인증, 메일, 예약된 작업, 디버깅과 관련된 로그 메시지를 제외한 대부분의 syslog 메시지를 저장하는 로그 파일은 무엇입니까?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 2. 다음 중 시스템의 보안 및 인증 작업에 대한 syslog 메시지를 저장하는 로그 파일은 무엇입니까?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 3. 다음 중 /var/log 디렉터리에서 syslog 메시지를 파일로 정렬하고 구성하는 서비스는 무엇입니까?
 - a. rsyslog
 - b. systemd-journald
 - c. auditd
 - d. tuned

- ▶ 4. 다음 중 인간이 읽을 수 있는 syslog 파일이 포함된 디렉터리를 무엇입니까?
 - a. /sys/kernel/debug
 - b. /var/log/journal
 - c. /run/log/journal
 - d. /var/log

- ▶ 5. 다음 중 메일 서버에 대한 syslog 메시지를 저장하는 파일은 무엇입니까?
 - a. /var/log/lastlog
 - b. /var/log/maillog
 - c. /var/log/tallylog
 - d. /var/log/boot.log

▶ 6. 다음 중 예약된 작업에 대한 syslog 메시지를 저장하는 파일은 무엇입니까?

- a. /var/log/cron
- b. /var/log/tallylog
- c. /var/log/spooler
- d. /var/log/secure

▶ 7. 다음 중 시스템 시작에 대한 콘솔 메시지를 저장하는 파일은 무엇입니까?

- a. /var/log/messages
- b. /var/log/cron
- c. /var/log/boot.log
- d. /var/log/secure

▶ 솔루션

시스템 로그 아키텍처 설명

다음 질문에 대해 올바른 답을 선택하십시오.

- ▶ 1. 다음 중 인증, 메일, 예약된 작업, 디버깅과 관련된 로그 메시지를 제외한 대부분의 syslog 메시지를 저장하는 로그 파일은 무엇입니까?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 2. 다음 중 시스템의 보안 및 인증 작업에 대한 syslog 메시지를 저장하는 로그 파일은 무엇입니까?
 - a. /var/log/maillog
 - b. /var/log/boot.log
 - c. /var/log/messages
 - d. /var/log/secure

- ▶ 3. 다음 중 /var/log 디렉터리에서 syslog 메시지를 파일로 정렬하고 구성하는 서비스는 무엇입니까?
 - a. rsyslog
 - b. systemd-journald
 - c. auditd
 - d. tuned

- ▶ 4. 다음 중 인간이 읽을 수 있는 syslog 파일이 포함된 디렉터리를 무엇입니까?
 - a. /sys/kernel/debug
 - b. /var/log/journal
 - c. /run/log/journal
 - d. /var/log

- ▶ 5. 다음 중 메일 서버에 대한 syslog 메시지를 저장하는 파일은 무엇입니까?
 - a. /var/log/lastlog
 - b. /var/log/maillog
 - c. /var/log/tallylog
 - d. /var/log/boot.log

▶ 6. 다음 중 예약된 작업에 대한 syslog 메시지를 저장하는 파일은 무엇입니까?

- a. /var/log/cron
- b. /var/log/tallylog
- c. /var/log/spooler
- d. /var/log/secure

▶ 7. 다음 중 시스템 시작에 대한 콘솔 메시지를 저장하는 파일은 무엇입니까?

- a. /var/log/messages
- b. /var/log/cron
- c. /var/log/boot.log
- d. /var/log/secure

syslog 파일 검토

목표

문제를 해결하거나 시스템 상태를 검토하기 위해 관련 syslog 파일의 이벤트를 해석합니다.

시스템에 이벤트 기록

많은 프로그램이 syslog 프로토콜을 사용하여 이벤트를 시스템 로그에 기록합니다. 각 로그 메시지는 기능(메시지를 생성하는 하위 시스템) 및 우선순위(메시지 심각도)에 따라 분류됩니다.

다음 표에는 표준 syslog 기능이 나와 있습니다.

syslog 기능 개요

Code	기능	기능 설명
0	kern	커널 메시지
1	user	사용자 수준 메시지
2	mail	메일 시스템 메시지
3	daemon	시스템 데몬 메시지
4	auth	인증 및 보안 메시지
5	syslog	내부 syslog 메시지
6	lpr	프린터 메시지
7	뉴스	네트워크 뉴스 메시지
8	uucp	UUCP 프로토콜 메시지
9	cron	시계 데몬 메시지
10	authpriv	비 시스템 권한 부여 메시지
11	ftp	FTP 프로토콜 메시지
16~23	local0~local7	사용자 지정 로컬 메시지

다음 표에는 표준 syslog 우선 순위가 내림차순으로 나와 있습니다.

syslog 우선 순위 개요

Code	우선 순위	우선 순위 설명
0	emerg	시스템을 사용할 수 없음

Code	우선 순위	우선 순위 설명
1	alert	즉각적인 조치 필요
2	crit	심각한 상태
3	err	심각하지 않은 오류 상태
4	경고	경고 상태
5	notice	정상이지만 중요한 이벤트
6	info	정보 이벤트
7	debug	디버깅 수준의 메시지

rsyslog 서비스는 로그 메시지의 기능 및 우선 순위를 사용하여 메시지를 어떻게 처리할지 결정합니다. 규칙은 **/etc/rsyslog.conf** 파일과 **/etc/rsyslog.d** 디렉터리의 파일 (.conf 확장자)에서 이 기능과 우선 순위를 구성합니다. 소프트웨어 패키지는 **/etc/rsyslog.d** 디렉터리에 적절한 파일을 설치하여 규칙을 추가할 수 있습니다.

syslog 메시지를 정렬하는 방법을 제어하는 각 규칙에 해당하는 줄이 구성 파일 중 하나에 있습니다. 각 줄의 왼쪽은 규칙과 일치하는 syslog 메시지의 기능 및 우선 순위를 나타냅니다. 각 줄의 오른쪽은 로그 메시지를 저장하거나 메시지를 전달할 파일을 나타냅니다. 별표(*)는 모든 값과 일치하는 와일드카드입니다.

예를 들어 **/etc/rsyslog.d** 파일의 다음 줄은 임의 우선 순위로 **authpriv** 기능에 전송된 메시지를 **/var/log/secure** 파일에 기록합니다.

```
authpriv.*          /var/log/secure
```

로그 메시지가 **rsyslog.conf** 파일에서 두 개 이상의 규칙과 일치하는 경우가 있습니다. 이 경우 하나의 메시지가 두 개 이상의 로그 파일에 저장됩니다. 우선 순위 필드의 **none** 키워드는 저장되는 메시지를 제한하기 위해 표시된 기능에 대한 메시지가 지정된 파일에 저장되지 않음을 나타냅니다.

syslog 메시지를 파일에 기록하는 대신 로그인한 모든 사용자의 터미널로 출력할 수도 있습니다. **rsyslog.conf** 파일에는 우선 순위가 **emerg**인 모든 syslog 메시지를 로그인한 모든 사용자의 터미널로 출력할 수 있는 설정이 있습니다.

rsyslog 서비스의 샘플 규칙

```
##### RULES #####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
#kern.*                      /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none           /var/log/messages

# The authpriv file has restricted access.
authpriv.*                         /var/log/secure

# Log all the mail messages in one place.
```

```

mail.*                                     -/var/log/maillog

# Log cron stuff
cron.*                                      /var/log/cron

# Everybody gets emergency messages
.emerg                                       :omusrmsg:

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                /var/log/spooler

# Save boot messages also to boot.log
local7.*                                      /var/log/boot.log

```



참고

syslog 하위 시스템에는 이 과정의 범위를 벗어나는 다양한 기능이 있습니다. 자세한 내용은 `rsyslog.conf(5)` 도움말 페이지 및 `/usr/share/doc/rsyslog/html/index.html`에서 `rsyslog-doc` 패키지가 제공하는 광범위한 HTML 설명서를 참조하십시오.

로그 파일 순환

`logrotate` 명령은 `/var/log` 디렉터리의 공간을 너무 많이 차지하지 않도록 로그 파일을 순환합니다. 로그 파일이 순환되면 순환 날짜를 나타내는 확장자로 파일 이름이 변경됩니다. 예를 들어 이전 `/var/log/messages` 파일이 2022년 3월 20일에 순환되면 `/var/log/messages-20220320` 파일로 이름이 변경됩니다. 이전 로그 파일이 순환되면 로그 파일을 생성하고 로그 파일을 작성한 서비스에 알립니다.

일반적으로 4주 기간 중에 순환된 후 디스크 공간을 확보하기 위해 가장 오래된 로그 파일부터 삭제됩니다. 예약된 작업은 `logrotate` 명령을 매일 실행하여 로그 파일의 순환 요구 사항을 확인합니다. 대부분의 로그 파일은 매주 순환됩니다. `logrotate` 명령은 일부 로그 파일을 더 빠르게 또는 더 느리게 순환하거나 특정 크기에 도달할 때 순환합니다.

syslog 항목 분석

로그 메시지는 로그 파일 시작 부분에 가장 오래된 메시지가 있고 끝에 최신 메시지가 있는 상태로 시작됩니다. `rsyslog` 서비스는 표준 형식을 사용하여 로그 파일에 항목을 기록합니다. 다음 예제는 `/var/log/secure` 로그 파일의 로그 메시지의 구조를 설명합니다.

```

Mar 20 20:11:48 localhost sshd[1433]: Failed password for student from 172.25.0.10
port 59344 ssh2

```

- **Mar 20 20:11:48** : 로그 항목의 타임스탬프를 기록합니다.
- **localhost** : 로그 메시지를 전송하는 호스트입니다.
- **sshd[1433]** : 로그 메시지를 전송한 프로그램 또는 프로세스 이름 및 PID 번호입니다.
- **Failed password for ...** : 전송된 메시지입니다.

로그 이벤트 모니터링

로그 파일에서 이벤트를 모니터링하면 문제를 재현하는 데 유용합니다. `tail -f /path/to/file` 명령은 지정된 파일의 마지막 10줄을 출력하며, 파일에 새로 작성된 줄을 계속 출력합니다.

예를 들어 실패한 로그인 시도를 모니터링하려면 한 터미널에서 **tail** 명령을 실행한 다음, 다른 터미널에서 사용자가 시스템에 로그인을 시도하는 동안 **root** 사용자로 **ssh** 명령을 실행합니다.

첫 번째 터미널에서 **tail** 명령을 실행합니다.

```
[root@host ~]# tail -f /var/log/secure
```

두 번째 터미널에서 **ssh** 명령을 실행합니다.

```
[root@host ~]# ssh root@hosta
root@hosta's password: redhat
...output omitted...
[root@hostA ~]#
```

로그 메시지는 첫 번째 터미널에 표시됩니다.

```
...output omitted...
Mar 20 09:01:13 host sshd[2712]: Accepted password for root from 172.25.254.254
port 56801 ssh2
Mar 20 09:01:13 host sshd[2712]: pam_unix(sshd:session): session opened for user
root by (uid=0)
```

수동으로 syslog 메시지 전송

logger 명령은 **rsyslog** 서비스에 메시지를 전송할 수 있습니다. 기본적으로 **logger** 명령은 **-p** 옵션으로 달리 지정하지 않는 한, 우선 순위가 **notice** 인 사용자 유형(**user.notice**)에 메시지를 전송합니다. **rsyslog** 서비스 구성에 대한 변경 사항을 테스트하는 데 유용합니다.

/var/log/boot.log 로그 파일에 기록되는 **rsyslog** 서비스에 메시지를 전송하려면 다음 **logger** 명령을 실행합니다.

```
[root@host ~]# logger -p local7.notice "Log entry created on host"
```



참조

logger(1), **tail(1)**, **rsyslog.conf(5)** 및 **logrotate(8)** 도움말 페이지

rsyslog 도움말

- **/usr/share/doc/rsyslog/html/index.html** 패키지에 의해 **rsyslog-doc** 제공

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/assembly_troubleshooting-problems-using-log-files_configuring-basic-system-settings

에서 Troubleshooting Problems Using Log Files를 참조하십시오.

▶ 연습 가이드

syslog 파일 검토

이 연습에서는 새 파일에 특정 로그 메시지를 작성하도록 **rsyslog** 서비스를 재구성합니다.

결과

- 우선 순위가 **debug**인 모든 로그 메시지를 **/var/log/messages-debug** 로그 파일에 작성하도록 **rsyslog** 서비스를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-syslog
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **/etc/rsyslog.d/debug.conf** 구성 파일을 변경하여 서비스와 관련된, 우선순위가 **debug** 이상인 모든 메시지를 새 **/var/log/messages-debug** 로그 파일에 기록하도록 **servera** 시스템의 **rsyslog** 서비스를 구성합니다.

- 2.1. 우선순위가 **debug** 이상인 모든 로그 메시지를 **/var/log/messages-debug** 로그 파일로 리디렉션하는 데 필요한 항목을 사용하여 **/etc/rsyslog.d/debug.conf** 파일을 생성합니다.

```
*.debug /var/log/messages-debug
```

이 구성 줄은 우선순위 수준이 **debug** 이상인 모든 기능의 syslog 메시지를 기록합니다.

- 구성 줄의 기능 필드에 있는 와일드카드(*)는 기능 또는 로그 메시지를 나타냅니다.
 - rsyslog** 서비스는 일치하는 메시지를 **/var/log/messages-debug** 로그 파일에 씁니다.
- 2.2. **rsyslog** 서비스를 다시 시작합니다.

```
[root@servera ~]# systemctl restart rsyslog
```

- ▶ 3. 우선 순위가 **debug**인 모든 로그 메시지가 **/var/log/messages-debug** 로그 파일에 표시되는지 확인합니다.

- 3.1. 우선 순위가 **debug**인 **user** 유형의 로그 메시지를 생성합니다.

```
[root@servera ~]# logger -p user.debug "Debug Message Test"
```

- 3.2. **/var/log/messages-debug** 로그 파일의 마지막 로그 메시지 10개를 보고 다른 로그 메시지 중 **Debug Message Test** 메시지가 표시되는지 확인합니다.

```
[root@servera ~]# tail /var/log/messages-debug
Feb 13 18:22:38 servera systemd[1]: Stopping System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25176]: [origin software="rsyslogd"
swVersion="8.37.0-9.el8" x-pid="25176" x-info="http://www.rsyslog.com"] exiting
on signal 15.
Feb 13 18:22:38 servera systemd[1]: Stopped System Logging Service.
Feb 13 18:22:38 servera systemd[1]: Starting System Logging Service...
Feb 13 18:22:38 servera rsyslogd[25410]: environment variable TZ is not set, auto
correcting this to TZ=/etc/localtime [v8.37.0-9.el8 try http://www.rsyslog.com/
e/2442 ]
Feb 13 18:22:38 servera systemd[1]: Started System Logging Service.
Feb 13 18:22:38 servera rsyslogd[25410]: [origin software="rsyslogd"
swVersion="8.37.0-9.el8" x-pid="25410" x-info="http://www.rsyslog.com"] start
Feb 13 18:27:58 servera root[25416]: Debug Message Test
```

- 3.3. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-syslog
```

이것으로 섹션을 완료합니다.

시스템 저널 항목 검토

목표

문제를 해결하거나 시스템 상태를 검토하기 위해 시스템 저널의 로그 항목을 찾아 해석합니다.

시스템 저널에서 이벤트 찾기

systemd-journald 서비스는 저널이라는 인덱싱된 구조적 바이너리 파일에 로깅 데이터를 저장합니다. 이 데이터에는 로그 이벤트에 대한 추가 정보가 포함됩니다. 예를 들어 **syslog** 이벤트의 경우 이 정보에 원본 메시지의 우선순위와 기능이 포함됩니다. 기능은 메시지를 시작한 프로세스를 추적하기 위해 **syslog** 서비스에서 할당하는 값입니다.



중요

Red Hat Enterprise Linux에서는 기본적으로 메모리 기반 **/run/log** 디렉터리에 시스템 저널이 저장됩니다. 시스템을 종료하면 **/run/log** 디렉터리의 내용이 손실됩니다. **journald** 디렉터리를 영구 위치로 변경할 수 있습니다. 이 내용에 대해서는 이 장의 뒷부분에서 설명합니다.

저널에서 로그 메시지를 검색하려면 **journalctl** 명령을 사용합니다. **journalctl** 명령을 사용하여 저널의 모든 메시지를 보거나 옵션과 기준에 따라 특정 이벤트를 검색할 수 있습니다. **root**로 명령을 실행하면 저널에 대한 전체 액세스 권한을 갖게 됩니다. 일반 사용자도 **journalctl** 명령을 사용할 수 있지만 특정 메시지가 표시되지 않도록 시스템에서 제한합니다.

```
[root@host ~]# journalctl
...output omitted...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on PipeWire Multimedia System Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Starting Create User's Volatile Files and Directories...
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Listening on D-Bus User Message Bus Socket.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Sockets.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Finished Create User's Volatile Files and Directories.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Basic System.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started User Manager for UID 0.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Reached target Main User Target.
Mar 15 04:42:16 host.lab.example.com systemd[2127]: Startup finished in 90ms.
Mar 15 04:42:16 host.lab.example.com systemd[1]: Started Session 6 of User root.
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session opened for user root(uid=0) by (uid=0)
Mar 15 04:42:17 host.lab.example.com systemd[1]: Starting Hostname Service...
Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.
lines 1951-2000 (END) q
```

journalctl 명령은 중요한 로그 메시지를 강조 표시합니다. 우선순위가 **notice** 또는 **warning** 인 메시지는 굵은 텍스트로 표시되고, 우선 순위가 **error** 이상인 메시지는 빨간색 텍스트로 표시됩니다.

저널을 성공적으로 사용하는 핵심은 관련 출력만 표시하도록 저널 검색을 제한하는 것입니다.

기본적으로 **journalctl** 명령의 **-n** 옵션은 마지막 10개 로그 항목을 표시합니다. 표시할 로그 항목 수를 지정하는 선택적 인수로 로그 항목 수를 조정할 수 있습니다. 예를 들어 마지막 5개 로그 항목을 검토하려는 경우 다음 **journalctl** 명령을 실행할 수 있습니다.

```
[root@host ~]# journalctl -n 5
Mar 15 04:42:17 host.lab.example.com systemd[1]: Started Hostname Service.
Mar 15 04:42:47 host.lab.example.com systemd[1]: systemd-hostnamed.service: Deactivated successfully.
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Created slice User Background Tasks Slice.
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Starting Cleanup of User's Temporary Files and Directories...
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's Temporary Files and Directories.
```

tail 명령과 유사하게 **journalctl** 명령의 **-f** 옵션은 시스템 저널의 마지막 10줄을 출력하며, 저널에 추가되는 새 저널 항목을 계속 출력합니다. **journalctl** 명령의 **-f** 옵션을 종료하려면 **Ctrl+C** 키 조합을 사용합니다.

```
[root@host ~]# journalctl -f
Mar 15 04:47:33 host.lab.example.com systemd[2127]: Finished Cleanup of User's Temporary Files and Directories.
Mar 15 05:01:01 host.lab.example.com CROND[2197]: (root) CMD (run-parts /etc/cron.hourly)
Mar 15 05:01:01 host.lab.example.com run-parts[2200]: (/etc/cron.hourly) starting Anacron
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Anacron started on 2022-03-15
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.daily' in 29 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.weekly' in 49 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Will run job `cron.monthly' in 69 min.
Mar 15 05:01:01 host.lab.example.com anacron[2208]: Jobs will be executed sequentially
Mar 15 05:01:01 host.lab.example.com run-parts[2210]: (/etc/cron.hourly) finished Anacron
Mar 15 05:01:01 host.lab.example.com CROND[2196]: (root) CMDEND (run-parts /etc/cron.hourly)
^C
[root@host ~]#
```

문제를 해결하려면 저널 항목의 우선순위로 저널 출력을 필터링하는 것이 좋습니다. **journalctl** 명령의 **-p** 옵션은 지정된 우선순위 수준(이름 또는 번호) 이상의 저널 항목을 표시합니다. **journalctl** 명령은 **debug, info, notice, warning, err, crit, alert, emerg** 우선 순위 수준을 오름차순으로 처리합니다.

예를 들어 다음 **journalctl** 명령을 실행하여 우선순위가 **err** 이상인 저널 항목을 표시합니다.

```
[root@host ~]# journalctl -p err
Mar 15 04:22:00 host.lab.example.com pipewire-pulse[1640]: pw.conf: execvp error
  'pactl': No such file or direct
Mar 15 04:22:17 host.lab.example.com kernel: Detected CPU family 6 model 13
  stepping 3
Mar 15 04:22:17 host.lab.example.com kernel: Warning: Intel Processor - this
  hardware has not undergone testing by Red Hat and might not be certif>
Mar 15 04:22:20 host.lab.example.com smartd[669]: DEVICESCAN failed: glob(3)
  aborted matching pattern /dev/disks/disc*
Mar 15 04:22:20 host.lab.example.com smartd[669]: In the system's table of devices
  NO devices found to scan
```

journalctl 명령의 **-u** 옵션과 유닛 이름을 사용하여 지정된 systemd 유닛에 대한 메시지를 표시할 수 있습니다.

```
[root@host ~]# journalctl -u sshd.service
May 15 04:30:18 host.lab.example.com systemd[1]: Starting OpenSSH server daemon...
May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on 0.0.0.0 port
  22.
May 15 04:30:18 host.lab.example.com sshd[1142]: Server listening on :: port 22.
May 15 04:30:18 host.lab.example.com systemd[1]: Started OpenSSH server daemon.
May 15 04:32:03 host.lab.example.com sshd[1796]: Accepted publickey for user1 from
  172.25.250.254 port 43876 ssh2: RSA SHA256:1UGy...>
May 15 04:32:03 host.lab.example.com sshd[1796]: pam_unix(sshd:session): session
  opened for user user1(uid=1000) by (uid=0)
May 15 04:32:26 host.lab.example.com sshd[1866]: Accepted publickey for user2
  from ::1 port 36088 ssh2: RSA SHA256:M8ik...
May 15 04:32:26 host.lab.example.com sshd[1866]: pam_unix(sshd:session): session
  opened for user user2(uid=1001) by (uid=0)
lines 1-8/8 (END) q
```

특정 이벤트를 찾을 때는 특정 시간 범위로 출력을 제한할 수 있습니다. **journalctl** 명령에는 특정 시간 범위로 출력을 제한하기 위한 두 가지 옵션이 있습니다. **--since** 및 **--until** 옵션입니다. 두 옵션 모두 "YYYY-MM-DD hh:mm:ss" 형식(옵션의 공백을 유지하려면 큰따옴표 사용 필수)의 시간 인수를 사용합니다.

시간 인수를 생략하면 **journalctl** 명령은 하루가 00:00:00에 시작된다고 가정합니다. 일 인수를 생략하면 명령은 현재 날짜를 가정합니다. 두 옵션 모두 날짜와 시간 필드를 비롯하여 **yesterday**, **today**, **tomorrow**를 유효한 인수로 사용합니다.

예를 들어 다음 **journalctl** 명령을 실행하여 오늘 레코드의 모든 저널 항목을 표시합니다.

```
[root@host ~]# journalctl --since today
...output omitted...
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Session 8 of User
  student.
Mar 15 05:04:20 host.lab.example.com sshd[2255]: pam_unix(sshd:session): session
  opened for user student(uid=1000) by (uid=0)
Mar 15 05:04:20 host.lab.example.com systemd[1]: Starting Hostname Service...
Mar 15 05:04:20 host.lab.example.com systemd[1]: Started Hostname Service.
Mar 15 05:04:50 host.lab.example.com systemd[1]: systemd-hostnamed.service:
  Deactivated successfully.
```

```
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Starting Mark boot as
successful...
Mar 15 05:06:33 host.lab.example.com systemd[2261]: Finished Mark boot as
successful.
lines 1996-2043/2043 (END) q
```

다음 `journalctl` 명령을 실행하여 2022-03-11 20:30:00 부터 2022-03-14 10:00:00까지의 모든 저널 항목을 표시합니다.

```
[root@host ~]# journalctl --since "2022-03-11 20:30" --until "2022-03-14 10:00"
...output omitted...
```

현재 기준 상대 시간 이후의 모든 항목을 지정할 수도 있습니다. 예를 들어 최근 1시간 동안의 모든 항목을 지정하려면 다음 명령을 사용할 수 있습니다.

```
[root@host ~]# journalctl --since "-1 hour"
...output omitted...
```



참고

`--since` 및 `--until` 옵션을 사용하여 더 정교한 다른 시간 사양을 사용할 수 있습니다. 몇 가지 예를 보려면 `systemd.time(7)` 도움말 페이지를 참조하십시오.

자세한 출력을 켜면 저널에 표시되는 내용 외에도 추가 로그 항목을 볼 수 있습니다. 표시되는 추가 필드를 사용하여 저널 쿼리 출력을 필터링할 수 있습니다. 자세한 출력은 저널의 특정 이벤트에 대한 복합 검색 결과 출력을 줄이는 데 유용합니다.

```
[root@host ~]# journalctl -o verbose
Tue 2022-03-15 05:10:32.625470 EDT [s=e7623387430b4c14b2c71917db58e0ee;i...]
 _BOOT_ID=beaadd6e5c5448e393ce716cd76229d4
 _MACHINE_ID=4ec03abd2f7b40118b1b357f479b3112
 PRIORITY=6
 SYSLOG_FACILITY=3
 SYSLOG_IDENTIFIER=systemd
 _UID=0
 _GID=0
 _TRANSPORT=journal
 _CAP_EFFECTIVE=1fffffff
 TID=1
 CODE_FILE=src/core/job.c
 CODE_LINE=744
 CODE_FUNC=job_emit_done_message
 JOB_RESULT=done
 _PID=1
 _COMM=systemd
 _EXE=/usr/lib/systemd/systemd
 _SYSTEMD_CGROUP=/init.scope
 _SYSTEMD_UNIT=init.scope
 _SYSTEMD_SLICE=--.slice
 JOB_TYPE=stop
 MESSAGE_ID=9d1aaa27d60140bd96365438aad20286
 _HOSTNAME=host.lab.example.com
```

```
_CMDLINE=/usr/lib/systemd/systemd --switched-root --system --deserialize 31
_SELINUX_CONTEXT=system_u:system_r:init_t:s0
UNIT=user-1000.slice
MESSAGE=Removed slice User Slice of UID 1000.
INVOCATION_ID=0e5efc1b4a6d41198f0cf02116ca8aa8
JOB_ID=3220
_SOURCE_REALTIME_TIMESTAMP=1647335432625470
lines 46560-46607/46607 (END) q
```

다음 목록은 특정 프로세스 또는 이벤트와 관련된 줄을 검색하는데 사용할 수 있는 시스템 저널의 일부 필드를 보여줍니다.

- _COMM - 명령 이름입니다.
- _EXE - 프로세스의 실행 파일 경로입니다.
- _PID - 프로세스의 PID입니다.
- _UID - 프로세스를 실행하는 사용자의 UID입니다.
- _SYSTEMD_UNIT - 프로세스를 시작한 **systemd** 유닛입니다.

journalctl 명령으로 여러 개의 시스템 저널 필드를 조합하여 세분화된 검색 쿼리를 만들 수 있습니다. 예를 들어 다음 **journalctl** 명령은 PID가 2110인 프로세스의 **sshd.service** **systemd** 유닛과 관련된 모든 저널 항목을 보여줍니다.

```
[root@host ~]# journalctl _SYSTEMD_UNIT=sshd.service _PID=2110
Mar 15 04:42:16 host.lab.example.com sshd[2110]: Accepted
    publickey for root from 172.25.250.254 port 46224 ssh2: RSA
    SHA256:1UGybTe52L2jzEJa1HLVKn9QUCKrTv3ZzxMjol1Fro
Mar 15 04:42:16 host.lab.example.com sshd[2110]: pam_unix(sshd:session): session
    opened for user root(uid=0) by (uid=0)
```



참고

저널 필드의 목록은 **systemd.journal-fields(7)** 도움말 페이지를 참조하십시오.



참조

journalctl(1), **systemd.journal-fields(7)** 및 **systemd.time(7)** 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

에서 Red Hat Enterprise Linux 9 Configuring Basic System Settings 가이드의 Troubleshooting Problems Using Log Files 섹션을 참조하십시오.

▶ 연습 가이드

시스템 저널 항목 검토

이 연습에서는 시스템 저널에서 특정 기준과 일치하는 이벤트를 기록하는 항목을 검색합니다.

결과

- 시스템 저널에서 다른 기준에 따라 이벤트를 기록하는 항목을 검색합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-systemd
```

지침

- ▶ 1. **workstation** 시스템에서 **student** 사용자로 **servera** 시스템에 대한 SSH 세션을 엽니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **journalctl** 명령의 **_PID=1** 옵션을 사용하여 **servera** 시스템의 **systemd** PID 1 프로세스에서 시작된 로그 이벤트만 표시합니다. **journalctl** 명령을 종료하려면 **q** 키를 누릅니다. 다음 출력은 하나의 예이며 시스템에 따라 다를 수 있습니다.

```
[student@servera ~]$ journalctl _PID=1
Mar 15 04:21:14 localhost systemd[1]: Finished Load Kernel Modules.
Mar 15 04:21:14 localhost systemd[1]: Finished Setup Virtual Console.
Mar 15 04:21:14 localhost systemd[1]: dracut ask for additional cmdline parameters
was skipped because all trigger condition checks failed.
Mar 15 04:21:14 localhost systemd[1]: Starting dracut cmdline hook...
Mar 15 04:21:14 localhost systemd[1]: Starting Apply Kernel Variables...
lines 1-5 q
[student@servera ~]$
```

- ▶ 3. **journalctl** 명령의 **_UID=81** 옵션을 사용하여 **servera** 시스템의 UID 81 시스템 서비스에서 시작된 모든 로그 이벤트를 표시합니다.

```
[student@servera ~]$ journalctl _UID=81
Mar 15 04:21:17 servera.lab.example.com dbus-broker-lau[727]: Ready
```

- ▶ 4. `journalctl` 명령의 `-p warning` 옵션을 사용하여 `servera` 시스템에서 우선순위가 `warning` 이상인 로그 이벤트를 표시합니다.

```
[student@servera ~]$ journalctl -p warning
Mar 15 04:21:14 localhost kernel: wait_for_initramfs() called before
rootfs_initcalls
Mar 15 04:21:14 localhost kernel: ACPI: PRMT not present
Mar 15 04:21:14 localhost kernel: acpi PNP0A03:00: fail to add MMCONFIG
information, can't access extended PCI configuration space under this bridge.
Mar 15 04:21:14 localhost kernel: device-mapper: core: CONFIG_IMA_DISABLE_HTABLE
is disabled. Duplicate IMA measurements will not be recorded in the IMA log.
...output omitted...
Mar 15 04:21:18 servera.lab.example.com NetworkManager[769]: <warn>
[1647332478.5504] device (eth0): mtu: failure to set IPv6 MTU
Mar 15 04:21:27 servera.lab.example.com chrony[751]: System clock wrong by
-0.919695 seconds
Mar 15 04:22:34 servera.lab.example.com chrony[751]: System clock wrong by
0.772805 seconds
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error:
kex_exchange_identification: Connection closed by remote host
lines 1-19/19 (END) q
[student@servera ~]$
```

- ▶ 5. `servera` 시스템에서 현재 시간부터 지난 10분 동안 기록된 모든 로그 이벤트를 표시합니다.

```
[student@servera ~]$ journalctl --since "-10min"
Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly' started
Mar 15 05:40:01 servera.lab.example.com anacron[1092]: Job `cron.weekly'
terminated
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: error:
kex_exchange_identification: Connection closed by remote host
Mar 15 05:41:11 servera.lab.example.com sshd[1104]: Connection closed by
172.25.250.9 port 45370
Mar 15 05:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for student
from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFct
+wowZLNzNlBT0
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Created slice User Slice of
UID 1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Runtime
Directory /run/user/1000...
Mar 15 05:41:14 servera.lab.example.com systemd-logind[739]: New session 1 of user
student.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Finished User Runtime
Directory /run/user/1000.
Mar 15 05:41:14 servera.lab.example.com systemd[1]: Starting User Manager for UID
1000...
...output omitted...
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Sockets.
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped target Timers.
```

```
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Mark boot as
successful after the user session has run 2 minutes.
Mar 15 05:44:56 servera.lab.example.com systemd[1109]: Stopped Daily Cleanup of
User's Temporary Directories.
lines 1-48 q
[student@servera ~]$
```

- ▶ 6. `journalctl` 명령의 `--since` 및 `_SYSTEMD_UNIT="sshd.service"` 옵션을 사용하여 `servera` 시스템에서 오늘 아침 **09:00:00** 이후 `sshd` 서비스에서 기록된 모든 로그 이벤트를 표시합니다.



참고

온라인 강의실은 일반적으로 UTC 시간대로 실행됩니다. 로컬 시간대로 오전 9시에 시작되는 결과를 얻으려면 UTC에서의 오프셋 양만큼 `--since` 값을 조정합니다. 또는 로컬 시간을 무시하고 값으로 9:00을 사용하여 `servera` 시간대로 9:00 이후에 발생한 저널 항목을 찾습니다.

```
[student@servera ~]$ journalctl --since 9:00:00 _SYSTEMD_UNIT="sshd.service"
Mar 15 09:41:14 servera.lab.example.com sshd[1105]: Accepted publickey for student
from 172.25.250.9 port 45372 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFct
+wowZLNzNlBT0
Mar 15 09:41:15 servera.lab.example.com sshd[1105]: pam_unix(sshd:session):
session opened for user student(uid=1000) by (uid=0)
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: Accepted publickey for student
from 172.25.250.9 port 45374 ssh2: RSA SHA256:M8ikhcEDm2tQ95Z0o7ZvufqEixCFct
+wowZLNzNlBT0
Mar 15 09:44:56 servera.lab.example.com sshd[1156]: pam_unix(sshd:session):
session opened for user student(uid=1000) by (uid=0)
```

- ▶ 7. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-systemd
```

이것으로 섹션을 완료합니다.

시스템 저널 보존

목표

서버가 재부팅될 때 이벤트 레코드를 보존하도록 시스템 저널을 구성합니다.

시스템 저널 스토리지

기본적으로 Red Hat Enterprise Linux 9에서는 `/run/log` 디렉터리에 시스템 저널을 저장하며, 재부팅 후 시스템에서 시스템 저널을 지웁니다. 재부팅 후에도 저널이 유지되도록 `/etc/systemd/journald.conf` 파일에서 `systemd-journald` 서비스의 구성 설정을 변경할 수 있습니다.

`/etc/systemd/journald.conf` 파일의 `Storage` 매개 변수는 시스템 저널을 일시적으로 저장할지 또는 재부팅 후에도 유지되도록 저장할지 정의합니다. 다음과 같이 이 매개 변수를 `persistent`, `volatile`, `auto` 또는 `none`으로 설정합니다.

- **persistent**: 재부팅 후에도 지속되는 `/var/log/journal` 디렉터리에 저널을 저장합니다. `/var/log/journal` 디렉터리가 없는 경우 `systemd-journald` 서비스에서 생성합니다.
- **volatile**: 일시적인 `/run/log/journal` 디렉터리에 저널을 저장합니다. `/run` 파일 시스템은 일시적이고 런타임 메모리에만 존재하므로 시스템 저널을 포함하여 이 파일 시스템의 데이터는 재부팅 후 유지되지 않습니다.
- **auto**: `/var/log/journal` 디렉터리가 있으면 `systemd-journald` 서비스는 영구저장장치를 사용하고, 없으면 일시적 스토리지를 사용합니다. `Storage` 매개 변수를 설정하지 않으면 이 동작이 기본값입니다.
- **none**: 스토리지를 사용하지 않습니다. 시스템에서 모든 로그를 삭제하지만 계속해서 로그를 전달할 수 있습니다.

영구 시스템 저널의 장점은 부팅할 때 즉시 과거 데이터를 사용할 수 있다는 것입니다. 그러나 영구 저널도 모든 데이터가 시스템에 영원히 유지되는 것은 아닙니다. 저널에는 매달 실행되는 기본 로그 로테이션 메커니즘이 있습니다. 또한 시스템에서 저널에 사용한 공간은 해당 저널이 있는 파일 시스템의 10% 이하여야 하고, 저널에 사용 가능한 공간은 파일 시스템의 15% 이상이어야 합니다. `/etc/systemd/journald.conf` 구성 파일에서 런타임 및 영구 저널의 이러한 값을 둘 다 수정할 수 있습니다.

`systemd-journald` 프로세스는 시작될 때 저널 크기에 대한 현재 제한을 기록됩니다. 다음 명령 출력에는 현재 크기 제한을 반영하는 저널 항목이 표시되어 있습니다.

```
[user@host ~]$ journalctl | grep -E 'Runtime Journal|System Journal'
Mar 15 04:21:14 localhost systemd-journald[226]: Runtime Journal (/run/log/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.
Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: Runtime Journal (/run/log/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 113.3M, 105.3M free.
Mar 15 04:21:19 host.lab.example.com systemd-journald[719]: System Journal (/run/log/journal/4ec03abd2f7b40118b1b357f479b3112) is 8.0M, max 4.0G, 4.0G free.
```

**참고**

이전 **grep** 명령에서 세로 막대(|) 기호는 or 연산자 역할을 합니다. 즉, **grep** 명령은 **journalctl** 명령 출력의 **Runtime Journal** 문자열이나 **System Journal** 문자열이 포함된 모든 줄을 찾습니다. 이 명령은 일시적(**Runtime**) 저널 저장소와 영구(**System**) 저널 저장소에 대한 현재 크기 제한을 가져옵니다.

영구 시스템 저널 구성

재부팅 후에도 시스템 저널이 영구적으로 보존되도록 **systemd-journald** 서비스를 구성하기 위해 다음을 수행합니다.

- **/var/log/journal** 디렉터리를 생성합니다.

```
[root@host ~]# mkdir /var/log/journal
```

- **/etc/systemd/journald.conf** 파일에서 **Storage** 매개 변수를 **persistent** 값으로 설정합니다. 선택한 텍스트 편집기를 수퍼유저로 실행하여 **/etc/systemd/journald.conf** 파일을 편집합니다.

```
[Journal]
Storage=persistent
...output omitted...
```

- **systemd-journald** 서비스를 재시작하여 구성 변경 사항을 적용합니다.

```
[root@host ~]# systemctl restart systemd-journald
```

systemd-journald 서비스가 성공적으로 재시작되면 서비스가 **/var/log/journal** 디렉터리의 하위 디렉터리를 생성합니다. **/var/log/journal** 디렉터리의 하위 디렉터리에는 16진 문자로 된 긴 이름이 있고 확장자가 **.journal**인 파일이 포함되어 있습니다. **.journal** 바이너리 파일은 인덱싱된 구조적 저널 항목을 저장합니다.

```
[root@host ~]# ls /var/log/journal
4ec03abd2f7b40118b1b357f479b3112
[root@host ~]# ls /var/log/journal/4ec03abd2f7b40118b1b357f479b3112
system.journal user-1000.journal
```

재부팅 후에도 시스템 저널이 유지되는 경우, **journalctl** 명령 출력에 현재 시스템 부팅 및 이전 시스템 부팅의 항목이 포함됩니다. 출력을 특정 시스템 부팅으로 제한하려면 **-b** 명령의 **journalctl** 옵션을 사용합니다. 다음 **journalctl** 명령은 첫 번째 시스템 부팅의 항목만 검색합니다.

```
[root@host ~]# journalctl -b 1
...output omitted...
```

다음 **journalctl** 명령은 두 번째 시스템 부팅의 항목만 검색합니다. 인수는 시스템을 두 번 이상 재부팅한 경우에만 의미가 있습니다.

```
[root@host ~]# journalctl -b 2
...output omitted...
```

--list-boots 옵션을 사용하여 **journalctl** 명령이 인식하는 시스템 부팅 이벤트를 표시할 수 있습니다.

```
[root@host ~]# journalctl --list-boots
-6 27de... Wed 2022-04-13 20:04:32 EDT-Wed 2022-04-13 21:09:36 EDT
-5 6a18... Tue 2022-04-26 08:32:22 EDT-Thu 2022-04-28 16:02:33 EDT
-4 e2d7... Thu 2022-04-28 16:02:46 EDT-Fri 2022-05-06 20:59:29 EDT
-3 45c3... Sat 2022-05-07 11:19:47 EDT-Sat 2022-05-07 11:53:32 EDT
-2 dfae... Sat 2022-05-07 13:11:13 EDT-Sat 2022-05-07 13:27:26 EDT
-1 e754... Sat 2022-05-07 13:58:08 EDT-Sat 2022-05-07 14:10:53 EDT
 0 ee2c... Mon 2022-05-09 09:56:45 EDT-Mon 2022-05-09 12:57:21 EDT
```

다음 **journalctl** 명령은 현재 시스템 부팅의 항목만 검색합니다.

```
[root@host ~]# journalctl -b
...output omitted...
```



참고

영구 저널과 시스템 간 충돌을 디버깅할 때는 일반적으로 충돌이 발생하기 전 재부팅까지로 저널 쿼리를 제한해야 합니다. **journalctl** 명령의 **-b** 옵션에 음수를 사용하여 출력에 포함할 이전 시스템 부팅 수를 나타낼 수 있습니다. 예를 들어 **journalctl -b -1** 명령은 출력을 이전 부팅만으로 제한합니다.



참조

systemd-journald.conf(5), systemd-journald(8) 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_basic_system_settings/index#troubleshooting-problems-using-log-files_getting-started-with-system-administration

에서 Red Hat Enterprise Linux 9 Configuring Basic System Settings 가이드의 Troubleshooting Problems Using Log Files 섹션을 참조하십시오.

▶ 연습 가이드

시스템 저널 보존

이 연습에서는 재부팅 후 데이터를 보존하도록 시스템 저널을 구성합니다.

결과

- 재부팅 후 데이터를 보존하도록 시스템 저널을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-preserve
```

지침

- ▶ 1. **workstation** 시스템에서 **servera** 시스템에 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 수퍼유저 권한으로 **/var/log/journal** 디렉터리가 없는지 확인합니다. **ls** 명령을 사용하여 **/var/log/journal** 디렉터리의 콘텐츠를 표시합니다. **sudo** 명령을 사용하여 **student** 사용자 권한을 승격합니다. 메시지가 표시되면 **student** 암호를 사용합니다.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
ls: cannot access '/var/log/journal': No such file or directory
```

/var/log/journal 디렉터리가 없으므로 재부팅 후 **systemd-journald** 서비스에서 로그 데이터를 보존하지 않습니다.

- ▶ 3. 재부팅 후에도 저널을 보존하도록 **servera** 시스템의 **systemd-journald** 서비스를 구성합니다.

- 3.1. **/var/log/journal** 디렉터리를 생성합니다.

```
[student@servera ~]$ sudo mkdir /var/log/journal
```

- 3.2. **/etc/systemd/journald.conf** 파일에서 **Storage=auto** 줄의 주석 처리를 제거하고 **Storage** 매개 변수를 **persistent** 값으로 설정합니다. **sudo vim /etc/systemd/journald.conf** 명령을 사용하여 구성 파일을 편집할 수 있습니다. **vim** 편집기 명령 모드에서 **/Storage=auto**를 입력하여 **Storage=auto** 줄을 검색할 수 있습니다.

```
...output omitted...
[Journal]
Storage=persistent
...output omitted...
```

3.3. **systemd-journald** 서비스를 재시작하여 구성 변경 사항을 적용합니다.

```
[student@servera ~]$ sudo systemctl restart systemd-journald.service
```

- ▶ 4. 재부팅 후에도 유지되도록 **servera** 시스템의 **systemd-journald** 서비스가 저널을 보존하는지 확인합니다.

4.1. **servera** 시스템을 재시작합니다.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

servera 시스템을 재시작하는 즉시 SSH 연결이 종료됩니다.

4.2. **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 4.3. 긴 16진 이름이 있는 하위 디렉터리가 **/var/log/journal** 디렉터리에 있는지 확인합니다. 해당 디렉터리에서 저널 파일을 찾을 수 있습니다. 시스템의 하위 디렉터리 이름은 다를 수 있습니다.

```
[student@servera ~]$ sudo ls /var/log/journal
[sudo] password for student: student
63b272eae8d5443ca7aaa5593479b25f
[student@servera ~]$ sudo ls /var/log/journal/63b272eae8d5443ca7aaa5593479b25f
system.journal user-1000.journal
```

4.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-preserve
```

이것으로 섹션을 완료합니다.

정확한 시간 유지 관리

목표

NTP(Network Time Protocol)를 사용하여 정확한 시간 동기화를 유지하고 시스템 저널 및 로그에서 기록하는 이벤트의 타임스탬프가 정확하도록 시간대를 구성합니다.

로컬 시계 및 시간대 관리

시스템 시간 동기화는 여러 시스템의 로그 파일을 분석하는 데 중요합니다. 또한 일부 서비스가 정확하게 작동하려면 시간 동기화가 필요할 수 있습니다. 시스템은 네트워크 타임 프로토콜을 사용하여 인터넷을 통해 올바른 시간 정보를 제공하고 얻습니다. 시스템은 NTP 풀 프로젝트와 같은 공용 NTP 서비스로부터 정확한 시간 정보를 얻을 수 있습니다. 또 다른 옵션은 고품질 하드웨어 시계와 동기화하여 로컬 클라이언트에 정확한 시간을 제공하는 것입니다.

timedatectl 명령은 시스템의 현재 시간, 시간대 및 NTP 동기화 설정을 포함하여 현재 시간과 관련된 시스템 설정의 개요를 표시합니다.

```
[user@host ~]$ timedatectl
      Local time: Wed 2022-03-16 05:53:05 EDT
      Universal time: Wed 2022-03-16 09:53:05 UTC
            RTC time: Wed 2022-03-16 09:53:05
           Time zone: America/New_York (EDT, -0400)
     System clock synchronized: yes
          NTP service: active
            RTC in local TZ: no
```

timedatectl 명령의 **list-timezones** 옵션을 사용하여 시간대 데이터베이스를 표시할 수 있습니다.

```
[user@host ~]$ timedatectl list-timezones
Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Bamako
...output omitted...
```

IANA(Internet Assigned Numbers Authority)는 공용 시간대 데이터베이스를 제공하고, **timedatectl** 명령은 해당 데이터베이스를 기반으로 하여 시간대 이름을 지정합니다. IANA는 대륙 또는 대양과 일반적으로 (항상은 아님) 시간대 지역에서 가장 큰 도시를 기준으로 시간대 이름을 지정합니다. 예를 들어 미국 산지 시간대는 대부분 **America/Denver**입니다.

시간대 내의 일부 지역에는 다른 일광 절약 시간 규칙이 적용됩니다. 예를 들어 미국 애리조나주(미국 산지 시간)는 대부분 일광 절약 시간 조정을 변경하지 않으며 **America/Phoenix** 시간대에 속합니다.

tzselect 명령을 사용하여 올바른 시간대 이름을 식별합니다. 이 명령은 사용자에게 대화형으로 시스템 위치에 대한 질문을 하며 정확한 시간대 이름을 출력합니다. 시스템의 시간대 설정은 변경하지 않습니다.

root 사용자는 **timedatectl** 명령의 **set-timezone** 옵션을 사용하여 현재 시간대를 업데이트하고 룰 시스템 설정을 변경할 수 있습니다. 예를 들어 다음 **timedatectl** 명령은 현재 시간대를 **America/Phoenix**로 업데이트합니다.

```
[root@host ~]# timedatectl set-timezone America/Phoenix
[root@host ~]# timedatectl
    Local time: Wed 2022-03-16 03:05:55 MST
    Universal time: Wed 2022-03-16 10:05:55 UTC
        RTC time: Wed 2022-03-16 10:05:55
       Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
      NTP service: active
     RTC in local TZ: no
```



참고

서버 시간대를 UTC(협정 세계시)로 설정할 수 있습니다. **tzselect** 명령에는 UTC 시간대의 이름이 포함되어 있지 않습니다. **timedatectl set-timezone UTC** 명령을 사용하여 시스템의 현재 시간대를 **UTC**로 설정하십시오.

timedatectl 명령의 **set-time** 옵션을 사용하여 시스템의 현재 시간을 변경합니다. "YYYY-MM-DD hh:mm:ss" 형식으로 시간을 지정할 수 있으며, 여기서 날짜 또는 시간은 생략할 수 있습니다. 예를 들어 다음 **timedatectl** 명령은 시간을 **09:00:00**로 변경합니다.

```
[root@host ~]# timedatectl set-time 09:00:00
[root@host ~]# timedatectl
    Local time: Fri 2019-04-05 09:00:27 MST
    Universal time: Fri 2019-04-05 16:00:27 UTC
        RTC time: Fri 2019-04-05 16:00:27
       Time zone: America/Phoenix (MST, -0700)
System clock synchronized: yes
      NTP service: active
     RTC in local TZ: no
```



참고

이전 예제는 실패하고 "Failed to set time: Automatic time synchronization is enabled" 오류 메시지가 표시될 수 있습니다. 이 경우, 먼저 자동 시간 동기화를 비활성화한 후 이 참고 사항 뒤에 설명된 대로 날짜 또는 시간을 수동으로 설정합니다.

timedatectl 명령의 **set-ntp** 옵션은 자동 시간 조정을 위한 NTP 동기화를 활성화 또는 비활성화합니다. 이 옵션을 켜거나 끄려면 **true** 또는 **false** 인수가 필요합니다. 예를 들어 다음 **timedatectl** 명령은 NTP 동기화를 끕니다.

```
[root@host ~]# timedatectl set-ntp false
```

**참고**

Red Hat Enterprise Linux 9에서 **timedatectl set-ntp** 명령은 **chronyd** NTP 서비스가 활성화되었는지 여부를 조정합니다. 다른 Linux 배포판에서는 이 설정을 사용하여 다른 NTP 또는 SNTP(Simple Network Time Protocol) 서비스를 조정할 수 있습니다.

그래픽 GNOME **Settings** 애플리케이션에서와 같이 Red Hat Enterprise Linux에서 다른 유ти리티를 사용하여 NTP를 활성화하거나 비활성화하면 이 설정도 업데이트됩니다.

chronyd 서비스 구성 및 모니터링

chronyd 서비스는 구성된 NTP 서버와 동기화하여 일반적으로 부정확한 로컬 RTC(실시간 시계)를 추적합니다. 네트워크 연결을 사용할 수 없는 경우 **chronyd** 서비스는 RTC 시계 드리프트를 계산하고, **/etc/chrony.conf** 구성 파일에서 **driftfile** 값으로 지정된 파일에 기록합니다.

기본적으로 **chronyd** 서비스는 NTP 풀 프로젝트의 서버를 사용하여 시간을 동기화하며 추가 구성이 필요하지 않습니다. 격리된 네트워크에서 실행되는 시스템의 NTP 서버를 변경해야 할 수도 있습니다.

NTP 시간 소스의 stratum에 따라 품질이 결정됩니다. stratum은 시스템이 고성능 참조 클록과 얼마나 떨어져 있는지를 흡수를 가지고 결정합니다. 참조 클록은 **stratum 0** 시간 소스입니다. 참조 시계에 직접 연결된 NTP 서버는 **stratum 1** 시간 소스입니다. NTP 서버에서 시간을 동기화하는 시스템은 **stratum 2** 시간 소스입니다.

/etc/chrony.conf 구성 파일에는 server 및 peer의 두 가지 시간 범주를 선언할 수 있습니다. server는 로컬 NTP 서버보다 한 계층 높으며 peer는 동일한 계층 수준에 있습니다. **chronyd** 구성 파일에서 한 줄에 하나씩, 여러 개의 서버와 피어를 정의할 수 있습니다.

server 행의 첫 번째 인수는 NTP 서버의 IP 주소나 DNS 이름입니다. 서버 IP 주소나 이름을 따라가면 서버에 대한 일련의 옵션을 표시할 수 있습니다. **iburst** 옵션을 사용하는 것이 좋습니다. 그러면 **chronyd** 서비스가 시작된 후 보다 정확한 초기 시간 동기화를 위해 서비스에서 짧은 기간 동안 네 번 측정합니다. **chronyd** 구성 파일 옵션에 대한 자세한 내용을 보려면 **man 5 chrony.conf** 명령을 사용합니다.

예를 들어 **/etc/chrony.conf** 구성 파일에 다음 **server classroom.example.com iburst** 줄을 추가하면 **chronyd** 서비스는 **classroom.example.com** 서버를 NTP 시간 소스로 사용합니다.

```
# Use public servers from the pool.ntp.org project.
...output omitted...
server classroom.example.com iburst
...output omitted...
```

chronyd 서비스에 **classroom.example.com** 로컬 시간 소스를 지정한 후 서비스를 재시작합니다.

```
[root@host ~]# systemctl restart chronyd
```

chronyc 명령은 **chronyd** 서비스에 대한 클라이언트 역할을 합니다. NTP 동기화를 설정한 후 **chronyc sources** 명령을 통해 로컬 시스템에서 원활하게 NTP 서버를 사용하여 시스템 시계를 동기화하는지 확인합니다. 출력에 대한 추가 설명과 함께 보다 자세한 출력을 보려면 **chronyc sources -v** 명령을 사용합니다.

```
[root@host ~]# chronyc sources -v
... Source mode '^' = server, '=' = peer, '#' = local clock.
```

```

/ .- Source state '*' = current best, '+' = combined, '-' = not combined,
| /           'x' = may be in error, '~' = too variable, '?' = unusable.
||                               . - xxxx [ yyyy ] +/- zzzz
||   Reachability register (octal) -. |   |   |   | xxxx = adjusted offset,
||   Log2(Polling interval) --. |   |   |   | yyyy = measured offset,
||                           \   |   |   |   \ zzzz = estimated error.
||                           |   |   |
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
^* 172.25.254.254        3     6    17    26 +2957ns[+2244ns] +/- 25ms

```

S(소스 상태) 필드의 별표 문자(*)는 **chrony**d 서비스가 **classroom.example.com** 서버를 시간 소스로 사용하며 시스템이 현재 동기화되어 있는 NTP 서버임을 나타냅니다.



참조

[timedatectl\(1\)](#), [tzselect\(8\)](#), [chrony\(8\)](#), [chrony.conf\(5\)](#) 및 [chronyc\(1\)](#) 도움
말 페이지

NTP 풀 프로젝트

<http://www.ntppool.org/>

시간대 데이터베이스

<http://www.iana.org/time-zones>

▶ 연습 가이드

정확한 시간 유지 관리

이 연습에서는 서버의 시간대를 조정하고, 시스템 시계가 NTP 시간 소스와 동기화되도록 합니다.

결과

- 서버의 시간대를 변경합니다.
- NTP 시간 소스와 시간을 동기화하도록 서버를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-maintain
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 이 연습에서는 **servera** 시스템이 아이티로 재배치되었으며 시간대를 업데이트해야 한다고 가정합니다. **timedatectl** 명령을 사용하여 시간대를 업데이트하도록 **student** 사용자의 권한을 승격합니다.

2.1. 아이티에 적절한 시간대를 선택합니다.

```
[student@servera ~]$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
1) Africa
2) Americas
3) Antarctica
4) Asia
5) Atlantic Ocean
6) Australia
7) Europe
8) Indian Ocean
9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the timezone using the Posix TZ format.
#? 2
```

```
Please select a country whose clocks agree with yours.
1) Anguilla    19) Dominican Republic    37) Peru
2) Antigua & Barbuda    20) Ecuador    38) Puerto Rico
3) Argentina    21) El Salvador    39) St Barthelemy
4) Aruba    22) French Guiana    40) St Kitts & Nevis
5) Bahamas    23) Greenland    41) St Lucia
6) Barbados    24) Grenada    42) St Maarten (Dutch)
7) Belize    25) Guadeloupe    43) St Martin (French)
8) Bolivia    26) Guatemala    44) St Pierre & Miquelon
9) Brazil    27) Guyana    45) St Vincent
10) Canada    28) Haiti    46) Suriname
11) Caribbean NL    29) Honduras    47) Trinidad & Tobago
12) Cayman Islands    30) Jamaica    48) Turks & Caicos Is
13) Chile    31) Martinique    49) United States
14) Colombia    32) Mexico    50) Uruguay
15) Costa Rica    33) Montserrat    51) Venezuela
16) Cuba    34) Nicaragua    52) Virgin Islands (UK)
17) Curaçao    35) Panama    53) Virgin Islands (US)
18) Dominica    36) Paraguay
#? 28
```

The following information has been given:

Haiti

Therefore TZ='America/Port-au-Prince' will be used.

Selected time is now: Wed Mar 16 07:10:35 EDT 2022.

Universal Time is now: Wed Mar 16 11:10:35 UTC 2022.

Is the above information OK?

1) Yes

2) No

#? 1

You can make this change permanent for yourself by appending the line

TZ='America/Port-au-Prince'; export TZ

to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you can use the /usr/bin/tzselect command in shell scripts:

America/Port-au-Prince

2.2. servera 시스템의 시간대를 America/Port-au-Prince로 업데이트합니다.

```
[student@servera ~]$ sudo timedatectl set-timezone \
America/Port-au-Prince
[sudo] password for student: student
```

2.3. 시간대를 America/Port-au-Prince로 올바르게 설정했는지 확인합니다.

```
[student@servera ~]$ timedatectl
    Local time: Wed 2022-03-16 07:13:25 EDT
    Universal time: Wed 2022-03-16 11:13:25 UTC
        RTC time: Wed 2022-03-16 11:13:24
      Time zone: America/Port-au-Prince (EDT, -0400)
System clock synchronized: no
          NTP service: inactive
    RTC in local TZ: no
```

- ▶ 3. 시스템 시간을 NTP 시간 소스인 `classroom.example.com` 서버와 동기화하도록 `servera` 시스템의 `chronyd` 서비스를 구성합니다.

- 3.1. `/etc/chrony.conf` 구성 파일을 편집하여 `classroom.example.com` 서버를 NTP 시간 소스로 지정합니다. 다음 출력은 구성 파일에 추가할 구성 줄을 보여줍니다. 여기에는 초기 시간 동기화 속도를 높이기 위한 `iburst` 옵션이 포함되어 있습니다.

```
...output omitted...
server classroom.example.com iburst
...output omitted...
```

- 3.2. `servera` 시스템에서 시간 동기화를 활성화합니다. 이 명령은 `/etc/chrony.conf` 구성 파일의 설정을 사용하여 NTP 서버를 활성화합니다. 현재 시스템에 설치된 서비스에 따라 명령이 `chronyd` 또는 `ntpd` 서비스를 활성화할 수 있습니다.

```
[student@servera ~]$ sudo timedatectl set-ntp true
```

- ▶ 4. `servera` 시스템 구성이 강의실 환경의 `classroom.example.com` 시간 소스와 동기화되는지 확인합니다.

- 4.1. `servera` 시스템에서 시간 동기화가 활성화되었는지 확인합니다.



참고

출력에 시간이 동기화되지 않은 것으로 표시되면 몇 초 동안 기다린 다음 `timedatectl` 명령을 재실행합니다. 시간 설정과 시간 소스를 동기화하는 데 몇 초가 걸립니다.

```
[student@servera ~]$ timedatectl
    Local time: Wed 2022-03-16 07:24:13 EDT
    Universal time: Wed 2022-03-16 11:24:13 UTC
        RTC time: Wed 2022-03-16 11:24:13
      Time zone: America/Port-au-Prince (EDT, -0400)
System clock synchronized: yes
          NTP service: active
    RTC in local TZ: no
```

- 4.2. `servera` 시스템이 현재 시간 설정을 `classroom.example.com` 시간 소스와 동기화하는지 확인합니다.

출력에서 **classroom.example.com** NTP 시간 소스의 소스 상태(S) 필드에 별표 문자(*) 가 표시됩니다. 별표는 로컬 시스템 시간이 NTP 시간 소스와 성공적으로 동기화되었음을 나타냅니다.

```
[student@servera ~]$ chronyc sources -v

    .-- Source mode '^' = server, '=' = peer, '#' = local clock.
    / .. Source state '*' = current best, '+' = combined, '-' = not combined,
    | /           'x' = may be in error, '~' = too variable, '?' = unusable.
    ||           'x' = may be in error, '~' = too variable, '?' = unusable.
    ||           Reachability register (octal) -.          .- xxxx [ yyyy ] +/- zzzz
    ||           Log2(Polling interval) --.      |          | xxxx = adjusted offset,
    ||           \           |          |          | yyyy = measured offset,
    ||           |           |          |          | zzzz = estimated error.
    ||           |           |          \
MS Name/IP address      Stratum Poll Reach LastRx Last sample
=====
^* 172.25.254.254        2     6   377    33    +84us[ +248us] +/-  21ms
```

4.3. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-maintain
```

이것으로 섹션을 완료합니다.

▶ 랩

로그 분석 및 저장

이 실습에서는 기존 서버의 시간대를 변경하고 인증 실패와 관련된 모든 이벤트에 대해 새 로그 파일을 구성합니다.

결과

- 기존 서버의 시간대를 업데이트합니다.
- 인증 실패에 대한 모든 메시지를 저장할 새 로그 파일을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-review
```

지침

- student** 사용자로 **serverb** 시스템에 로그인합니다.
- serverb** 시스템이 자메이카로 재배치되었으며 시간대를 **America/Jamaica**로 업데이트해야 한다고 가정합니다. 적절한 시간대를 올바르게 설정했는지 확인합니다.
- serverb** 시스템에서 이전 30분 동안 기록된 로그 이벤트를 봅니다.
- /etc/rsyslog.d/auth-errors.conf** 파일을 생성합니다. **Logging test authpriv.alert** 메시지를 **/var/log/auth-errors** 파일에 쓰도록 **rsyslog** 서비스를 구성합니다. **authpriv** 기능과 **alert** 우선 순위를 사용합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade logs-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

로그 분석 및 저장

이 실습에서는 기존 서버의 시간대를 변경하고 인증 실패와 관련된 모든 이벤트에 대해 새 로그 파일을 구성합니다.

결과

- 기존 서버의 시간대를 업데이트합니다.
- 인증 실패에 대한 모든 메시지를 저장할 새 로그 파일을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start logs-review
```

지침

- student** 사용자로 **serverb** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- serverb** 시스템이 자메이카로 재배치되었으며 시간대를 **America/Jamaica**로 업데이트해야 한다고 가정합니다. 적절한 시간대를 올바르게 설정했는지 확인합니다.

- 2.1. 자메이카에 적절한 시간대를 선택합니다.

```
[student@serverb ~]$ tzselect
Please identify a location so that time zone rules can be set correctly.
Please select a continent, ocean, "coord", or "TZ".
1) Africa
2) Americas
3) Antarctica
4) Asia
5) Atlantic Ocean
6) Australia
7) Europe
8) Indian Ocean
9) Pacific Ocean
10) coord - I want to use geographical coordinates.
11) TZ - I want to specify the timezone using the Posix TZ format.
#? 2
Please select a country whose clocks agree with yours.
```

1) Anguilla	19) Dominican Republic	37) Peru
2) Antigua & Barbuda	20) Ecuador	38) Puerto Rico
3) Argentina	21) El Salvador	39) St Barthelemy
4) Aruba	22) French Guiana	40) St Kitts & Nevis
5) Bahamas	23) Greenland	41) St Lucia
6) Barbados	24) Grenada	42) St Maarten (Dutch)
7) Belize	25) Guadeloupe	43) St Martin (French)
8) Bolivia	26) Guatemala	44) St Pierre & Miquelon
9) Brazil	27) Guyana	45) St Vincent
10) Canada	28) Haiti	46) Suriname
11) Caribbean NL	29) Honduras	47) Trinidad & Tobago
12) Cayman Islands	30) Jamaica	48) Turks & Caicos Is
13) Chile	31) Martinique	49) United States
14) Colombia	32) Mexico	50) Uruguay
15) Costa Rica	33) Montserrat	51) Venezuela
16) Cuba	34) Nicaragua	52) Virgin Islands (UK)
17) Curaçao	35) Panama	53) Virgin Islands (US)
18) Dominica	36) Paraguay	
#? 30		

The following information has been given:

Jamaica

Therefore TZ='America/Jamaica' will be used.

Selected time is now: Wed Mar 16 07:17:15 EST 2022.

Universal Time is now: Wed Mar 16 12:17:15 UTC 2022.

Is the above information OK?

1) Yes

2) No

#? 1

You can make this change permanent for yourself by appending the line

TZ='America/Jamaica'; export TZ

to the file '.profile' in your home directory; then log out and log in again.

Here is that TZ value again, this time on standard output so that you can use the /usr/bin/tzselect command in shell scripts:

America/Jamaica

2.2. **serverb** 서버의 시간대를 America/Jamaica로 업데이트하도록 **student** 사용자 권한을 승격합니다.

```
[student@serverb ~]$ sudo timedatectl set-timezone America/Jamaica
[sudo] password for student: student
```

2.3. 시간대를 America/Jamaica로 성공적으로 설정했는지 확인합니다.

```
[student@serverb ~]$ timedatectl
    Local time: Wed 2022-03-16 07:18:40 EST
    Universal time: Wed 2022-03-16 12:18:40 UTC
        RTC time: Wed 2022-03-16 12:18:40
     Time zone: America/Jamaica (EST, -0500)
System clock synchronized: yes
          NTP service: active
    RTC in local TZ: no
```

3. serverb 시스템에서 이전 30분 동안 기록된 로그 이벤트를 봅니다.

3.1. 저널 항목을 볼 시간 프레임을 결정합니다.

```
[student@serverb ~]$ date
Wed Mar 16 07:19:29 AM EST 2022
[student@serverb ~]$ date -d "-30 minutes"
Wed Mar 16 06:49:38 AM EST 2022
```

3.2. serverb 시스템에서 이전 30분 동안 기록된 로그 이벤트를 봅니다.

```
[student@serverb ~]$ journalctl --since 06:49:00 --until 07:19:00
...output omitted...
Mar 16 07:10:58 localhost kernel: x86/PAT: Configuration [0-7]: WB WC UC- UC WB
    WP UC- WT
Mar 16 07:10:58 localhost kernel: found SMP MP-table at [mem
    0x000f5bd0-0x000f5bdf]
Mar 16 07:10:58 localhost kernel: Using GB pages for direct mapping
Mar 16 07:10:58 localhost kernel: RAMDISK: [mem 0x2e0d9000-0x33064fff]
Mar 16 07:10:58 localhost kernel: ACPI: Early table checksum verification disabled
Mar 16 07:10:58 localhost kernel: ACPI: RSDP 0x00000000000F5B90 000014 (v00
    BOCHS )
Mar 16 07:10:58 localhost kernel: ACPI: RSDT 0x000000007FFE12C4 00002C (v01 BOCHS
    BXPCRSDT 00000001 BXPC 00000001)
Mar 16 07:10:58 localhost kernel: ACPI: FACP 0x000000007FFE11D0 000074 (v01 BOCHS
    BXPCFACP 00000001 BXPC 00000001)
Mar 16 07:10:58 localhost kernel: ACPI: DSDT 0x000000007FFDFDC0 001410 (v01 BOCHS
    BXPCDSDT 00000001 BXPC 00000001)
lines 1-50/50 q
[student@serverb ~]$
```

4. /etc/rsyslog.d/auth-errors.conf 파일을 생성합니다. Logging test authpriv.alert 메시지를 /var/log/auth-errors 파일에 쓰도록 rsyslog 서비스를 구성합니다. authpriv 기능과 alert 우선 순위를 사용합니다.

4.1. /etc/rsyslog.d/auth-errors.conf 파일을 생성하고 새 /var/log/auth-errors 파일을 인증 및 보안 메시지의 대상으로 지정합니다.

```
authpriv.alert  /var/log/auth-errors
```

4.2. rsyslog 서비스를 재시작하여 구성 파일 변경 사항을 적용합니다.

```
[student@serverb ~]$ sudo systemctl restart rsyslog
```

- 4.3. `logger -p` 명령을 사용하여 `Logging test authpriv.alert` 메시지를 `/var/log/auth-errors` 파일에 작성합니다. `authpriv` 기능과 `alert` 우선 순위를 사용합니다.

```
[student@serverb ~]$ logger -p authpriv.alert "Logging test authpriv.alert"
```

- 4.4. `/var/log/auth-errors` 파일에 `Logging test authpriv.alert` 메시지를 포함하는 로그 항목이 있는지 확인합니다.

```
[student@serverb ~]$ sudo tail /var/log/auth-errors
Mar 16 07:25:12 serverb student[1339]: Logging test authpriv.alert
```

- 4.5. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade logs-review
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish logs-review
```

이것으로 섹션을 완료합니다.

요약

- **systemd-journald** 및 **rsyslog** 서비스는 로그 메시지를 검색하여 적절한 파일에 기록합니다.
- **/var/log** 디렉터리에는 로그 파일이 들어 있습니다.
- 로그 파일을 주기적으로 순환하면 파일 시스템 공간이 꽉 차는 것을 방지할 수 있습니다.
- **systemd** 저널은 일시적이며 재부팅 후에는 유지되지 않습니다.
- **chrony** 서비스를 사용하면 시간 설정을 시간 소스와 동기화할 수 있습니다.
- 위치에 따라 서버의 시간대를 업데이트할 수 있습니다.

13장

네트워킹 관리

목적

Red Hat EnterpriseLinux 서버의 네트워크 인터페이스 및 설정 구성

목표

- 명령줄 유ти리티를 사용하여 현재 네트워크 구성을 테스트 및 검사합니다.
- nmcli 명령을 사용하여 네트워크 설정 및 장치를 관리합니다.
- 구성 파일을 편집하여 네트워크 구성을 수정합니다.
- 서버의 정적 호스트 이름과 이름 확인을 구성하고 결과를 테스트합니다.

섹션

- 네트워크 구성 확인(안내에 따른 연습)
- 명령줄에서 네트워킹 구성(안내에 따른 연습)
- 네트워크 구성 파일 편집(안내에 따른 연습)
- 호스트 이름 및 이름 확인 구성(안내에 따른 연습)

랩

- 네트워킹 관리

네트워크 구성 유효성 검사

목표

명령줄 유ти리티를 사용하여 현재 네트워크 구성을 테스트 및 검사합니다.

네트워크 인터페이스 정보 수집

ip link 명령은 시스템에서 사용 가능한 모든 네트워크 인터페이스를 표시합니다. 다음 예제에서는 서버에 세 가지 네트워크 인터페이스가 있습니다. **lo**는 서버 자체에 연결된 루프백 장치이고 **ens3** 및 **ens4**는 두 개의 이더넷 인터페이스입니다.

```
[user@host ~]$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:00:1e brd ff:ff:ff:ff:ff:ff
```

네트워크 인터페이스를 올바르게 구성하려면 각 인터페이스가 연결된 네트워크를 알고 있어야 합니다. 카드 또는 서버에 물리적으로 출력되었거나 가상 시스템이고 구성 방법을 알고 있기 때문에 각 네트워크에 연결된 인터페이스의 MAC 주소를 얻을 수 있는 경우가 많습니다. 장치의 MAC 주소는 각 인터페이스에 대해 **link/ether** 다음에 표시됩니다. 따라서 MAC 주소 **52:54:00:00:00:0a**가 있는 네트워크 카드는 네트워크 인터페이스 **ens3**임을 알고 있습니다.

IP 주소 표시

ip 명령을 사용하여 장치 및 주소 정보를 확인합니다. 단일 네트워크 인터페이스에 여러 개의 IPv4 또는 IPv6 주소가 있을 수 있습니다.

```
[user@host ~]$ ip addr show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000①
    link/ether 52:54:00:00:00:0b brd ff:ff:ff:ff:ff:ff②
    inet 192.0.2.2/24 brd 192.0.2.255 scope global ens3③
        valid_lft forever preferred_lft forever
    inet6 2001:db8:0:1:5054:ff:fe00:b/64 scope global④
        valid_lft forever preferred_lft forever
    inet6 fe80::5054:ff:fe00:b/64 scope link⑤
        valid_lft forever preferred_lft forever
```

① 활성 인터페이스는 **UP**입니다.

② **link/ether** 문자열은 장치의 하드웨어(MAC) 주소를 지정합니다.

- ❸ **inet** 문자열은 IPv4 주소, 해당 네트워크 접두사 길이 및 범위를 표시합니다.
- ❹ **inet6** 문자열은 IPv6 주소, 해당 네트워크 접두사 길이 및 범위를 표시합니다. 이 주소는 글로벌 범위이며 일반적으로 사용됩니다.
- ❺ **inet6** 문자열은 인터페이스에 로컬 이더넷 링크의 통신에만 사용할 수 있는 링크 범위의 IPv6 주소가 있음을 표시합니다.

성능 통계 표시

ip 명령은 네트워크 성능에 대한 통계를 표시할 수도 있습니다. 각 네트워크 인터페이스의 카운터는 네트워크 문제가 있는지 확인할 수 있습니다. 카운터는 수신(RX) 및 전송(TX)된 패킷, 패킷 오류, 삭제된 패킷 수와 같은 통계를 기록합니다.

```
[user@host ~]$ ip -s link show ens3
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 52:54:00:00:00:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes   packets   errors   dropped overrun mcast
        269850      2931       0        0        0        0
        TX: bytes   packets   errors   dropped carrier collsns
        300556      3250       0        0        0        0
```

호스트 간 연결 확인

ping 명령은 연결을 테스트합니다. 이 명령은 전송된 패킷 수를 제한하는 옵션이 설정되지 않은 한, **Ctrl+c** 를 누를 때까지 계속 실행됩니다.

```
[user@host ~]$ ping -c3 192.0.2.254
PING 192.0.2.1 (192.0.2.254) 56(84) bytes of data.
64 bytes from 192.0.2.254: icmp_seq=1 ttl=64 time=4.33 ms
64 bytes from 192.0.2.254: icmp_seq=2 ttl=64 time=3.48 ms
64 bytes from 192.0.2.254: icmp_seq=3 ttl=64 time=6.83 ms

--- 192.0.2.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 3.485/4.885/6.837/1.424 ms
```

ping6 명령은 Red Hat Enterprise Linux의 **ping** 명령에 해당하는 IPv6 버전입니다. 두 명령의 차이점은 **ping6** 명령은 IPv6를 통해 통신하고 IPv6 주소를 사용한다는 것입니다.

```
[user@host ~]$ ping6 2001:db8:0:1::1
PING 2001:db8:0:1::1(2001:db8:0:1::1) 56 data bytes
64 bytes from 2001:db8:0:1::1: icmp_seq=1 ttl=64 time=18.4 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=2 ttl=64 time=0.178 ms
64 bytes from 2001:db8:0:1::1: icmp_seq=3 ttl=64 time=0.180 ms
^C
--- 2001:db8:0:1::1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 0.178/6.272/18.458/8.616 ms
[user@host ~]$
```

링크-로컬 주소 및 링크-로컬 all-nodes 멀티캐스트 그룹(**ff02::1**)을 ping하면 범위 영역 식별자(예: **ff02::1%ens3**)를 사용하여 사용할 네트워크 인터페이스를 명시적으로 지정해야 합니다. 이 네트워크 인터페이스를 생략하면 connect: Invalid argument 오류가 표시됩니다.

ping6 ff02::1 명령을 사용하여 로컬 네트워크에서 다른 IPv6 노드를 찾을 수 있습니다.

```
[user@host ~]$ ping6 ff02::1%ens4
PING ff02::1%ens4(fe02::1) 56 data bytes
64 bytes from fe80::78cf:ffff:fed2:f97b: icmp_seq=1 ttl=64 time=22.7 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=30.1 ms (DUP!)
64 bytes from fe80::78cf:ffff:fed2:f97b: icmp_seq=2 ttl=64 time=0.183 ms
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=2 ttl=64 time=0.231 ms (DUP!)
^C
--- ff02::1%ens4 ping statistics ---
2 packets transmitted, 2 received, +2 duplicates, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 0.183/13.320/30.158/13.374 ms
[user@host ~]$

[user@host ~]$ ping6 -c 1 fe80::f482:dbff:fe25:6a9f%ens4
PING fe80::f482:dbff:fe25:6a9f%ens4(fe80::f482:dbff:fe25:6a9f) 56 data bytes
64 bytes from fe80::f482:dbff:fe25:6a9f: icmp_seq=1 ttl=64 time=22.9 ms

--- fe80::f482:dbff:fe25:6a9f%ens4 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 22.903/22.903/22.903/0.000 ms
```

동일한 링크의 다른 호스트는 IPv6 링크-로컬 주소를 일반 주소처럼 사용할 수 있습니다.

```
[user@host ~]$ ssh fe80::f482:dbff:fe25:6a9f%ens4
user@fe80::f482:dbff:fe25:6a9f%ens4's password:
Last login: Thu Jun  5 15:20:10 2014 from host.example.com
[user@server ~]$
```

라우터 문제 해결

네트워크 라우팅은 복잡하며 트래픽이 예상대로 동작하지 않는 경우가 있습니다. 다양한 툴을 사용하여 라우터 문제를 진단할 수 있습니다.

라우팅 테이블 설명

ip 명령의 **route** 옵션을 사용하여 라우팅 정보를 표시합니다.

```
[user@host ~]$ ip route
default via 192.0.2.254 dev ens3 proto static metric 1024
192.0.2.0/24 dev ens3 proto kernel scope link src 192.0.2.2
10.0.0.0/8 dev ens4 proto kernel scope link src 10.0.0.11
```

10.0.0.0/8 네트워크를 대상으로 하는 모든 패킷은 **ens4** 장치를 통해 대상으로 직접 전송됩니다.

192.0.2.0/24 네트워크를 대상으로 하는 모든 패킷은 **ens3** 장치를 통해 대상으로 직접 전송됩니다. 기타 모든 패킷은 **192.0.2.254**에 있는 기본 라우터로 전송되며 장치 **ens3**를 통해서도 전송됩니다.

ip 명령의 **-6** 옵션을 사용하여 IPv6 라우팅 테이블을 표시합니다.

```
[user@host ~]$ ip -6 route
unreachable ::/96 dev lo metric 1024 error -101
unreachable ::ffff:0.0.0.0/96 dev lo metric 1024 error -101
2001:db8:0:1::/64 dev ens3 proto kernel metric 256
unreachable 2002:a00::/24 dev lo metric 1024 error -101
unreachable 2002:7f00::/24 dev lo metric 1024 error -101
unreachable 2002:a9fe::/32 dev lo metric 1024 error -101
unreachable 2002:ac10::/28 dev lo metric 1024 error -101
unreachable 2002:c0a8::/32 dev lo metric 1024 error -101
unreachable 2002:e000::/19 dev lo metric 1024 error -101
unreachable 3ffe:ffff::/32 dev lo metric 1024 error -101
fe80::/64 dev ens3 proto kernel metric 256
default via 2001:db8:0:1::ffff dev ens3 proto static metric 1024
```

1. **2001:db8:0:1::/64** 네트워크는 **ens3** 인터페이스를 사용합니다(해당 네트워크에 주소가 있을 수 있음).
2. **fe80::/64** 네트워크는 링크-로컬 주소를 위해 **ens3** 인터페이스를 사용합니다. 여러 개의 인터페이스가 있는 시스템에서는 각 링크-로컬 주소를 위한 각 인터페이스에 **fe80::/64** 네트워크의 경로가 있습니다.
3. IPv6 인터넷(**::/0** 네트워크)에 있는 모든 네트워크의 기본 경로는 **2001:db8:0:1::ffff** 네트워크의 라우터를 사용하여 **ens3** 장치를 사용하여 연결할 수 있습니다.

트래픽 경로 추적

여러 라우터를 통해 원격 호스트에 연결하는 네트워크 트래픽 경로를 추적하려면 **traceroute** 또는 **tracepath** 명령을 사용합니다. 두 명령은 라우터 중 하나 또는 중간 라우터와 관련된 문제를 식별할 수 있습니다. 두 명령 모두 기본적으로 UDP 패킷을 사용하여 경로를 추적하지만, 대부분 네트워크는 UDP 및 ICMP 트래픽을 차단합니다. **traceroute** 명령에는 UDP(기본값), ICMP(-I) 또는 TCP(-T) 패킷으로 경로를 추적할 수 있는 옵션이 있습니다. 일반적으로 **traceroute** 명령은 기본적으로 설치되어 있지 않습니다.

```
[user@host ~]$ tracepath access.redhat.com
...output omitted...
4: 71-32-28-145.rcmt.qwest.net          48.853ms asymm 5
5: dcp-brdr-04.inet.qwest.net           100.732ms asymm 7
6: 206.111.0.153.ptr.us.xo.net          96.245ms asymm 7
7: 207.88.14.162.ptr.us.xo.net          85.270ms asymm 8
8: ae1d0.cir1.atlanta6-ga.us.xo.net    64.160ms asymm 7
9: 216.156.108.98.ptr.us.xo.net         108.652ms
10: bu-ether13.atlngamq46w-bcr00.tbone.rr.com 107.286ms asymm 12
...output omitted...
```

tracepath 명령 출력의 각 줄은 소스와 최종 목적지 간에 패킷이 통과하는 라우터 또는 hop을 나타냅니다. 이 명령은 RTT(라운드 트립 타이밍) 및 MTU(최대 전송 유닛)의 변경 사항을 포함하여 사용할 수 있게 되는 각 hop에 대한 정보를 출력합니다. **asymm** 표시는 라우터에 도달한 트래픽이 해당 라우터에서 다른(비대칭) 경로로 반환되었음을 의미합니다. 여기서 이 라우터는 반환 트래픽이 아닌 아웃바운드 트래픽을 위한 것입니다.

tracepath6 및 **traceroute -6** 명령은 **tracepath** 및 **traceroute** 명령에 해당하는 IPv6 명령입니다.

```
[user@host ~]$ tracepath 2001:db8:0:2::451
 1?: [LOCALHOST]                                0.091ms pmtu 1500
 1: 2001:db8:0:1::ba                          0.214ms
 2: 2001:db8:0:1::1                           0.512ms
 3: 2001:db8:0:2::451                         0.559ms reached
Resume: pmtu 1500 hops 3 back 3
```

포트 및 서비스 문제 해결

TCP 서비스는 소켓을 통신용 엔드포인트로 사용하며 IP 주소, 프로토콜, 포트 번호로 이루어져 있습니다. 일반적으로 서비스는 표준 포트에서 수신 대기를 하는 반면 클라이언트는 사용 가능한 임의의 포트를 활용하여 표준 포트에 연결됩니다. 표준 포트의 잘 알려진 이름은 **/etc/services** 파일에 표시되어 있습니다.

ss 명령은 소켓 통계를 표시하는데 사용됩니다. **ss** 명령은 **net-tools** 패키지의 기존 **netstat** 툴을 대체합니다. 일부 시스템 관리자는 이 툴이 더 익숙할 수도 있지만 항상 설치되지는 않습니다.

```
[user@host ~]$ ss -ta
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN      0      128           *:sunrpc                  *:*  
①
LISTEN      0      128           *:ssh                     *:*  
②
LISTEN      0      100          127.0.0.1:smtp            :  
LISTEN      0      128           *:36889                  *:  

ESTAB       0      0             172.25.250.10:ssh        172.25.254.254:59392  
③
LISTEN      0      128           :::sunrpc                 ::*  
④
LISTEN      0      128           :::ssh                    ::*  
⑤
LISTEN      0      100          ::1:smtp                  ::*  

LISTEN      0      128           :::34946                  ::*
```

- ❶ ***:ssh**: SSH에 사용되는 포트가 모든 IPv4 주소에서 수신 대기합니다. 별표(*) 문자는 IPv4 주소 또는 포트를 참조할 때 모두를 나타냅니다.
- ❷ **127.0.0.1:smtp**: SMTP에 사용되는 포트가 **127.0.0.1** IPv4 루프백 인터페이스에서 수신 대기합니다.
- ❸ **172.25.250.10:ssh**: SSH 연결이 172.25.250.10 인터페이스에서 설정되었으며, 주소가 172.25.254.254인 시스템에서 시작됩니다.
- ❹ **::ssh**: SSH에 사용되는 포트가 모든 IPv6 주소에서 수신 대기합니다. 이중 콜론(::) 구문은 모든 IPv6 인터페이스를 나타냅니다.
- ❺ **::1:smtp**: SMTP에 사용되는 포트가 ::1 IPv6 루프백 인터페이스에서 수신 대기합니다.

ss 및 netstat에 대한 옵션

옵션	설명
-n	인터페이스 및 포트의 이름 대신 번호를 표시합니다.
-t	TCP 소켓을 표시합니다.
-u	UDP 소켓을 표시합니다.
-l	연결된 소켓만 표시합니다.

옵션	설명
-a	모든(연결 및 설정된) 소켓을 표시합니다.
-p	소켓을 사용하는 프로세스를 표시합니다.
-A inet	inet 주소 패밀리에 대한 활성 연결(수신 대기 소켓이 아닌)을 표시합니다. 즉, 로컬 UNIX 도메인 소켓을 무시합니다. ss 명령의 경우 IPv4 및 IPv6 연결이 둘 다 표시됩니다. netstat 명령의 경우 IPv4 연결만 표시됩니다. (netstat -A inet6 명령은 IPv6 연결을 표시하고, netstat -46 명령은 IPv4 및 IPv6를 동시에 표시합니다.)



참조

ip-link(8), ip-address(8), ip-route(8), ip(8), ping(8), tracepath(8), traceroute(8), ss(8) 및 **netstat(8)** 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index
에 있는 Configuring and Managing Networking Guide를 참조하십시오.

▶ 연습 가이드

네트워크 구성 유효성 검사

이 연습에서는 서버 중 하나의 네트워크 구성을 검사합니다.

결과

- 현재 네트워크 인터페이스 및 네트워크 주소를 식별합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-validate
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **servera**에 **student** 사용자로 로그인합니다. 시스템은 인증 및 **servera**에 대한 암호가 없는 액세스에 SSH 키를 사용하도록 구성되었습니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. 이더넷 주소 **52:54:00:00:fa:0a**와 연결된 네트워크 인터페이스 이름을 찾습니다. 이 이름을 기록하거나 기억하고 이를 사용하여 후속 명령에서 **enX** 자리 표시자를 대체합니다.



중요

네트워크 인터페이스 이름은 부팅하는 동안 버스 유형 및 장치 탐지 순서에 따라 결정됩니다. 네트워크 인터페이스 이름은 사용 중인 교육 과정 플랫폼 및 하드웨어에 따라 다릅니다.

시스템에서 **52:54:00:00:fa:0a** 이더넷 주소와 연결된 인터페이스 이름(예: **ens06** 또는 **en1p2**)을 찾습니다. 이 인터페이스 이름을 사용하여 이 연습에서 사용된 **enX** 자리 표시자를 대체합니다.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group
    default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
```

- ▶ 3. 모든 인터페이스의 현재 IP 주소 및 넷 마스크를 표시합니다.

```
[student@servera ~]$ ip -br addr
lo          UP      127.0.0.1/8 ::1/128
enX:        UP      172.25.250.10/24 fe80::3059:5462:198:58b2/64
```

- ▶ 4. enX 인터페이스에 대한 통계를 표시합니다.

```
[student@servera ~]$ ip -s link show enX
2: enX: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        RX: bytes   packets   errors   dropped overrun mcast
            89014225    168251      0       154418      0       0
        TX: bytes   packets   errors   dropped carrier collsns
            608808     6090      0       0       0       0
```

- ▶ 5. 경로 정보를 표시합니다.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev enX proto static metric 100
172.25.250.0/24 dev enX proto kernel scope link src 172.25.250.10 metric 100
```

- ▶ 6. 라우터에 액세스할 수 있는지 확인합니다.

```
[student@servera ~]$ ping -c3 172.25.250.254
PING 172.25.250.254 (172.25.250.254) 56(84) bytes of data.
64 bytes from 172.25.250.254: icmp_seq=1 ttl=64 time=0.196 ms
64 bytes from 172.25.250.254: icmp_seq=2 ttl=64 time=0.436 ms
64 bytes from 172.25.250.254: icmp_seq=3 ttl=64 time=0.361 ms

--- 172.25.250.254 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 49ms
rtt min/avg/max/mdev = 0.196/0.331/0.436/0.100 ms
```

- ▶ 7. 로컬 시스템과 classroom.example.com 시스템 사이의 모든 흡을 표시합니다.

```
[student@servera ~]$ tracepath classroom.example.com
1?: [LOCALHOST]                                     pmtu 1500
1:  bastion.lab.example.com                         0.337ms
1:  bastion.lab.example.com                         0.122ms
2:  172.25.254.254                                  0.602ms reached
Resume: pmtu 1500 hops 2 back 2
```

- ▶ 8. 로컬 시스템에서 연결된 TCP 소켓을 표시합니다.

```
[student@servera ~]$ ss -lt
State      Recv-Q Send-Q      Local Address:Port          Peer Address:Port
LISTEN      0      128          0.0.0.0:sunrpc        0.0.0.0:*
LISTEN      0      128          0.0.0.0:ssh          0.0.0.0:*
LISTEN      0      128          [::]:sunrpc        [::]:*
LISTEN      0      128          [::]:ssh          [::]:*
```

- ▶ 9. workstation 시스템에 student 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-validate
```

이것으로 섹션을 완료합니다.

명령줄에서 네트워킹 구성

목표

nmcli 명령을 사용하여 네트워크 설정 및 장치를 관리합니다.

NetworkManager 서비스 설명

NetworkManager 서비스는 시스템의 네트워크 설정을 모니터링하고 관리합니다. GNOME 그래픽 환경에서 알림 영역 애플릿은 **NetworkManager** 데몬으로부터 받은 네트워크 구성 및 상태 정보를 표시합니다. 명령줄 또는 그래픽 툴을 사용하여 **NetworkManager** 서비스와 상호 작용할 수 있습니다. 서비스 구성 파일은 `/etc/NetworkManager/system-connections/` 디렉터리에 저장됩니다.

NetworkManager 서비스는 네트워크 장치 및 연결을 관리합니다. 장치는 네트워크 트래픽을 위해 제공되는 물리 또는 가상 네트워크 인터페이스입니다. 연결에는 단일 네트워크 장치와 관련된 구성 설정이 있습니다. 연결을 네트워크 프로필이라고도 합니다. 각 연결에는 고유한 이름 또는 ID가 있어야 하며, 구성하는 장치 이름과 일치할 수 있습니다.

단일 장치에 여러 개의 연결 구성이 있을 수 있으며 구성 간에 전환할 수 있지만 장치당 하나의 연결만 활성화 할 수 있습니다. 예를 들어 랩탑 무선 장치에서 연결의 보안 작업 사이트에서 사용할 고정 IP 주소를 구성할 수 있지만, 가정에서 동일한 회사 네트워크에 액세스하기 위해 자동화된 주소 및 VPN(가상 사설 네트워크)을 사용하여 보조 연결을 구성할 수도 있습니다.



중요

Red Hat Enterprise Linux 8부터 **ifcfg** 형식 구성 파일과 `/etc/sysconfig/network-scripts/` 디렉터리는 더 이상 사용되지 않습니다. 이제 **NetworkManager**는 키-값 쌍 구조인 INI 스타일 키 파일 형식을 사용하여 속성을 구성합니다. **NetworkManager**는 `/etc/NetworkManager/system-connections/` 디렉터리에 네트워크 프로필을 저장합니다. 이전 버전과의 호환성을 위해 `/etc/sysconfig/network-scripts/` 디렉터리의 **ifcfg** 형식 연결은 계속 인식되고 로드됩니다.

네트워크 정보 보기

nmcli 유틸리티를 사용하여 명령줄에서 연결 파일을 생성하고 편집합니다. **nmcli device status** 명령을 실행하면 모든 네트워크 장치의 상태가 표시됩니다.

```
[user@host ~]$ nmcli dev status
DEVICE  TYPE      STATE       CONNECTION
eno1    ethernet  connected   eno1
ens3    ethernet  connected   static-ens3
eno2    ethernet  disconnected --
lo     loopback  unmanaged   --
```

**참고**

`nmcli` 오브젝트와 작업을 축약할 수 있습니다. 예를 들어 `nmcli device disconnect` 는 `nmcli dev disconnect`, `nmcli connection modify` 는 `nmcli con mod`로 축약할 수 있습니다. 약어는 최소 1자까지 짧아도 되지만 관리할 오브젝트를 고유하게 식별할 만한 문자를 사용해야 합니다.

`nmcli connection show` 명령을 실행하면 모든 연결 목록이 표시됩니다. 활성 연결만 표시하려면 `--active` 옵션을 사용합니다.

```
[user@host ~]$ nmcli con show
NAME      UUID                                  TYPE      DEVICE
eno2      ff9f7d69-db83-4fed-9f32-939f8b5f81cd 802-3-ethernet --
static-ens3 72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1      87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
[user@host ~]$ nmcli con show --active
NAME      UUID                                  TYPE      DEVICE
static-ens3 72ca57a2-f780-40da-b146-99f71c431e2b 802-3-ethernet ens3
eno1      87b53c56-1f5d-4a29-a869-8a7bdaf56dfa 802-3-ethernet eno1
```

네트워크 연결 추가

`nmcli connection add` 명령을 사용하여 네트워크 연결을 추가합니다. 추가된 네트워크 연결에 대한 데이터는 `/etc/NetworkManager/system-connections/` 디렉터리에 접미사가 `.nmconnection`인 파일로 저장됩니다.

다음 예에서는 `eno2` 네트워크 인터페이스에 대해 `ethernet` 유형의 `eno2` 연결을 추가합니다.

```
[root@host ~]# nmcli con add con-name eno2 \
type ethernet iface eno2
Connection 'eno2' (8159b66b-3c36-402f-aa4c-2ea933c7a5ce) successfully added
```

다음 예에서는 정적 IPv4 네트워크 설정을 사용하여 `eno3` 네트워크 인터페이스에 대해 `ethernet` 유형의 `eno3` 연결을 생성합니다. 이 명령은 네트워크 접두사가 `/24`이고 네트워크 게이트웨이가 `192.168.0.254`인 `192.168.0.5` IP 주소를 구성합니다. 추가하려는 연결 이름이 있으면 `nmcli connection add` 명령이 실패합니다.

```
[root@host ~]# nmcli con add con-name eno3 type ethernet iface eno3 \
ipv4.method manual ipv4.addresses 192.168.0.5/24 ipv4.gateway 192.168.0.254
```

다음 예제에서는 고정 IPv6 및 IPv4 주소를 사용하여 `eno4` 장치에 대한 `eno4` 연결을 생성합니다. 이 명령은 네트워크 접두사가 `/64`이고 기본 게이트웨이가 `2001:db8:0:1::1` 주소인 `2001:db8:0:1::c000:207` IPv6 주소를 구성합니다. 또한 네트워크 접두사가 `/24`이고 기본 게이트웨이가 `192.0.2.1` 주소인 `192.0.2.7` IPv4 주소를 구성합니다.

```
[root@host ~]# nmcli con add con-name eno4 type ethernet iface eno4 \
ipv6.addresses 2001:db8:0:1::c000:207/64 ipv6.gateway 2001:db8:0:1::1 \
ipv6.method manual ipv4.addresses 192.0.2.7/24 ipv4.gateway 192.0.2.1 \
ipv4.method manual
```

네트워크 연결 관리

nmcli connection up 명령은 바인딩된 장치에서 네트워크 연결을 활성화합니다. 네트워크 연결을 활성화하려면 장치 이름이 아닌 연결 이름이 필요합니다.

```
[user@host ~]$ nmcli con show
NAME           UUID                                  TYPE      DEVICE
static-ens3    72ca57a2-f780-40da-b146-99f71c431e2b  802-3-ethernet  --
static-ens5    87b53c56-1f5d-4a29-a869-8a7bdaf56dfa  802-3-ethernet  --
[root@host ~]# nmcli con up static-ens3
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

nmcli device disconnect 명령은 네트워크 장치의 연결을 끊고 연결을 종료합니다.

```
[root@host ~]# nmcli dev disconnect ens3
```



중요

nmcli device disconnect를 사용하여 네트워크 인터페이스의 트래픽을 중지하고 연결을 비활성화합니다.

대부분의 연결은 **autoconnect** 매개 변수를 활성화하므로 **nmcli connection down** 명령은 트래픽을 중지하는 데 효과적이지 않습니다. 연결이 비활성화된 경우에도 자동 연결은 장치가 작동하고 사용할 수 있게 되면 즉시 연결을 재활성화합니다. 자동 연결은 일시적인 네트워크 중단이 발생해도 연결을 유지 관리하므로 바람직한 동작입니다.

연결된 장치의 연결을 끊으면 장치가 다시 연결될 때까지 연결이 강제로 종료됩니다.

네트워크 연결 설정 업데이트

NetworkManager 서비스 연결에는 두 가지 종류의 설정이 있습니다. 정적 연결 속성은 관리자가 구성하며 `/etc/NetworkManager/system-connections/* .nmconnection` 구성 파일에 저장됩니다. 동적 연결 속성은 DHCP 서버에서 요청되며 영구적으로 저장되지 않습니다.

연결의 현재 설정을 표시하려면 **nmcli connection show** 명령을 사용합니다. 소문자로 된 설정은 관리자가 변경할 수 있는 정적 속성입니다. 대문자로 된 설정은 이 연결 인스턴스에서 일시적으로 사용되는 활성 설정입니다.

```
[root@host ~]# nmcli con show static-ens3
connection.id:                      static-ens3
connection.uuid:                     87b53c56-1f5d-4a29-a869-8a7bdaf56dfa
connection.interface-name:          --
connection.type:                    802-3-ethernet
connection.autoconnect:             yes
connection.timestamp:              1401803453
connection.read-only:               no
connection.permissions:            --
connection.zone:                   --
connection.master:                 --
connection.slave-type:              --
connection.secondaries:            --
connection.gateway-ping-timeout:   0
```

```

802-3-ethernet.port:          --
802-3-ethernet.speed:         0
802-3-ethernet.duplex:        --
802-3-ethernet.auto-negotiate: yes
802-3-ethernet.mac-address:   CA:9D:E9:2A:CE:F0
802-3-ethernet.cloned-mac-address: --
802-3-ethernet.mac-address-blacklist:
802-3-ethernet.mtu:           auto
802-3-ethernet.s390-subchannels:
802-3-ethernet.s390-nettype:   --
802-3-ethernet.s390-options:
ipv4.method:                  manual
ipv4.dns:                      192.168.0.254
ipv4.dns-search:               example.com
ipv4.addresses:                { ip = 192.168.0.2/24,
                                 gw = 192.168.0.254 }

ipv4.routes:
ipv4.ignore-auto-routes:       no
ipv4.ignore-auto-dns:          no
ipv4.dhcp-client-id:          --
ipv4.dhcp-send-hostname:       yes
ipv4.dhcp-hostname:            --
ipv4.never-default:           no
ipv4.may-fail:                 yes
ipv6.method:                  manual
ipv6.dns:                      2001:4860:4860::8888
ipv6.dns-search:               example.com
ipv6.addresses:                { ip = 2001:db8:0:1::7/64,
                                 gw = 2001:db8:0:1::1 }

ipv6.routes:
ipv6.ignore-auto-routes:       no
ipv6.ignore-auto-dns:          no
ipv6.never-default:           no
ipv6.may-fail:                 yes
ipv6.ip6-privacy:              -1 (unknown)
ipv6.dhcp-hostname:            --
...output omitted...

```

nmcli connection modify 명령을 사용하여 연결 설정을 업데이트합니다. 이러한 변경 사항은 /etc/NetworkManager/system-connections/name.nmconnection 파일에 저장됩니다. 사용 가능한 설정은 **nm-settings(5)** 도움말 페이지를 참조하십시오.

다음 명령을 사용하여 **192.0.2.2/24** IPv4 주소 및 **192.0.2.254** 기본 게이트웨이를 설정하도록 **static-ens3** 연결을 업데이트합니다. **nmcli** 명령의 **connection.autoconnect** 매개 변수를 사용하여 시스템 부팅 시 연결을 자동으로 활성화하거나 비활성화합니다.

```
[root@host ~]# nmcli con mod static-ens3 ipv4.addresses 192.0.2.2/24 \
               ipv4.gateway 192.0.2.254 connection.autoconnect yes
```

다음 명령을 사용하여 **2001:db8:0:1::a00:1/64** IPv6 주소 및 **2001:db8:0:1::1** 기본 게이트웨이를 설정하도록 **static-ens3** 연결을 업데이트합니다.

```
[root@host ~]# nmcli con mod static-ens3 ipv6.addresses 2001:db8:0:1::a00:1/64 \
    ipv6.gateway 2001:db8:0:1::1
```



중요

DHCP 연결 구성을 정적으로 변경하려면 `ipv4.method` 설정을 `auto` 또는 `dhcp`에서 `manual`로 업데이트합니다. IPv6 연결의 경우 `ipv6.method` 설정을 업데이트합니다. 메서드가 올바르게 설정되지 않은 경우 활성화될 때 연결이 중단되거나 불완전할 수 있습니다. 또는 구성된 고정 주소 외에도 DHCP 또는 SLAAC에서 주소를 얻을 수 있습니다.

일부 설정에는 여러 개의 값이 있을 수 있습니다. 설정 이름의 시작 부분에 더하기(+) 또는 빼기(-) 기호를 추가하여 특정 값을 목록에 추가하거나 연결 설정에서 삭제할 수 있습니다. 더하기 또는 빼기가 포함되지 않은 경우 지정한 값이 설정의 현재 목록을 대체합니다. 다음 예제에서는 2.2.2.2 DNS 서버를 `static-ens3` 연결에 추가합니다.

```
[root@host ~]# nmcli con mod static-ens3 +ipv4.dns 2.2.2.2
```

`/etc/NetworkManager/system-connections/`에서 연결의 구성 파일을 편집하여 네트워크 프로필을 수정할 수도 있습니다. `nmcli` 명령은 NetworkManager와 직접 통신하여 수정 사항을 즉시 구현하지만, NetworkManager에 구성 파일을 다시 로드하도록 요청할 때까지 연결 파일 편집 내용은 구현되지 않습니다. 수동 편집을 사용하면 복잡한 구성을 단계별로 생성한 다음, 준비되었을 때 최종 구성을 로드할 수 있습니다. 다음 예제에서는 모든 연결 프로필을 로드합니다.

```
[root@host ~]# nmcli con reload
```

다음 예제에서는 `/etc/NetworkManager/system-connections/eno2.nmconnection`에 있는 `eno2` 연결 프로필만 로드합니다.

```
[root@host ~]# nmcli con reload eno2
```

네트워크 연결 삭제

`nmcli connection delete` 명령은 시스템에서 연결을 삭제합니다. 이 명령은 장치의 연결을 끊고 연결 구성 파일을 제거합니다.

```
[root@host ~]# nmcli con del static-ens3
```

NetworkManager 설정 수정 권한

`root` 사용자는 `nmcli` 명령을 사용하여 네트워크 구성을 변경할 수 있습니다.

물리 또는 가상 콘솔에 로그인한 권한 없는 사용자도 대부분의 네트워크 구성을 변경할 수 있습니다. 사용자가 시스템 콘솔에 있는 경우 시스템은 사용자가 연결을 구성, 활성화 및 비활성화해야 하는 워크스테이션 또는 랩톱으로 사용되고 있을 가능성이 큽니다. `ssh`로 로그인한 권한 없는 사용자는 `root` 사용자로 전환하여 네트워크 설정을 변경해야 합니다.

`nmcli general permissions` 명령을 사용하여 현재 권한을 봅니다. 다음 예제에서는 `root` 사용자의 NetworkManager 권한을 표시합니다.

```
[root@host ~]# nmcli gen permissions
PERMISSION                                         VALUE
org.freedesktop.NetworkManager.checkpoint-rollback   yes
org.freedesktop.NetworkManager.enable-disable-connectivity-check   yes
org.freedesktop.NetworkManager.enable-disable-network   yes
org.freedesktop.NetworkManager.enable-disable-statistics   yes
org.freedesktop.NetworkManager.enable-disable-wifi   yes
org.freedesktop.NetworkManager.enable-disable-wimax   yes
org.freedesktop.NetworkManager.enable-disable-wwan   yes
org.freedesktop.NetworkManager.network-control   yes
org.freedesktop.NetworkManager.reload   yes
org.freedesktop.NetworkManager.settings.modify.global-dns   yes
org.freedesktop.NetworkManager.settings.modify.hostname   yes
org.freedesktop.NetworkManager.settings.modify.own   yes
org.freedesktop.NetworkManager.settings.modify.system   yes
org.freedesktop.NetworkManager.sleep-wake   yes
org.freedesktop.NetworkManager.wifi.scan   yes
org.freedesktop.NetworkManager.wifi.share.open   yes
org.freedesktop.NetworkManager.wifi.share.protected   yes
```

다음 예제에서는 사용자의 NetworkManager 권한을 표시합니다.

```
[user@host ~]$ nmcli gen permissions
PERMISSION                                         VALUE
org.freedesktop.NetworkManager.checkpoint-rollback   auth
org.freedesktop.NetworkManager.enable-disable-connectivity-check   no
org.freedesktop.NetworkManager.enable-disable-network   no
org.freedesktop.NetworkManager.enable-disable-statistics   no
org.freedesktop.NetworkManager.enable-disable-wifi   no
org.freedesktop.NetworkManager.enable-disable-wimax   no
org.freedesktop.NetworkManager.enable-disable-wwan   no
org.freedesktop.NetworkManager.network-control   auth
org.freedesktop.NetworkManager.reload   auth
org.freedesktop.NetworkManager.settings.modify.global-dns   auth
org.freedesktop.NetworkManager.settings.modify.hostname   auth
org.freedesktop.NetworkManager.settings.modify.own   auth
org.freedesktop.NetworkManager.settings.modify.system   auth
org.freedesktop.NetworkManager.sleep-wake   no
org.freedesktop.NetworkManager.wifi.scan   auth
org.freedesktop.NetworkManager.wifi.share.open   no
org.freedesktop.NetworkManager.wifi.share.protected   no
```

유용한 NetworkManager 명령

다음 표에는 이 섹션에서 설명하는 주요 `nmcli` 명령이 나와 있습니다.

명령	목적
<code>nmcli dev status</code>	모든 네트워크 인터페이스의 NetworkManager 상태를 표시합니다.
<code>nmcli con show</code>	모든 연결을 나열합니다.

명령	목적
<code>nmcli con show name</code>	연결 name의 현재 설정을 표시합니다.
<code>nmcli con add con-name name</code>	새 연결 프로필을 추가하고 이름을 지정합니다.
<code>nmcli con mod name</code>	연결 name을 수정합니다.
<code>nmcli con reload</code>	수동 파일 편집 후에 구성 파일을 다시 로드합니다.
<code>nmcli con up name</code>	연결 name을 활성화합니다.
<code>nmcli dev dis dev</code>	인터페이스 연결을 끊습니다. 현재 연결도 비활성화됩니다.
<code>nmcli con del name</code>	지정된 연결과 해당 구성 파일을 삭제합니다.



참조

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index#getting-started-with-nmcli_configuring-and-managing-networking
에서 Getting Started with nmcli 장을 참조하십시오.

`NetworkManager(8)`, `nmcli(1)`, `nmcli-examples(5)`, `nm-settings(5)`, `hostnamectl(1)`, `resolv.conf(5)`, `hostname(5)`, `ip(8)` 및 `ip-address(8)` 도움말 페이지

▶ 연습 가이드

명령줄에서 네트워킹 구성

이 연습에서는 **nmcli** 명령을 사용하여 네트워크 설정을 구성합니다.

결과

- 네트워크 연결 설정을 DHCP에서 정적으로 업데이트합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-configure
```

지침

- ▶ 1. **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. 네트워크 인터페이스 정보를 표시합니다.



중요

네트워크 인터페이스 이름은 부팅하는 동안 버스 유형 및 장치 탐지 순서에 따라 결정됩니다. 네트워크 인터페이스 이름은 사용 중인 과정 플랫폼 및 하드웨어에 따라 다를 수 있습니다.

시스템에서 이더넷 주소 **52:54:00:00:fa:0a**와 연결된 인터페이스 이름(예: **eth1**, **ens06** 또는 **enp0p2**)을 찾습니다. 이 인터페이스 이름을 사용하여 연습에 있는 **eth0** 자리 표시자를 대체합니다(다른 경우).

이더넷 주소 **52:54:00:00:fa:0a**와 연결된 네트워크 인터페이스 이름을 찾습니다. 이 이름을 기록하거나 기억하고 이를 사용하여 후속 명령에서 **eth0** 자리 표시자를 대체합니다.

```
[root@servera ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT
    group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        altname ens3
```

▶ 3. `nmcli` 명령을 사용하여 네트워크 설정을 봅니다.

3.1. `nmcli con show --active` 명령을 사용하여 활성 연결만 표시합니다.

네트워크 인터페이스 이름이 출력의 **DEVICE** 열 아래에 표시되고, 해당 장치에 대한 활성 연결 이름이 **NAME** 열 아래에 표시됩니다. 이 연습에서는 활성 연결이 **Wired connection 1**이라고 가정합니다. 활성 연결 이름이 다른 경우 이 연습의 나머지 부분에서는 **Wired connection 1** 대신 해당 이름을 사용합니다.

```
[root@servera ~]# nmcli con show --active
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ec3a15fb-2e26-3254-9433-90c66981e924  ethernet  eth0
```

3.2. 활성화된 연결의 모든 설정 내용을 표시합니다.

```
[root@servera ~]# nmcli con show "Wired connection 1"
connection.id:                  Wired connection 1
connection.uuid:                 ec3a15fb-2e26-3254-9433-90c66981e924
connection.stable-id:            --
connection.type:                802-3-ethernet
connection.interface-name:       eth0
connection.autoconnect:         yes
...output omitted...
ipv4.method:                    manual
ipv4.dns:                       172.25.250.220
ipv4.dns-search:                lab.example.com,example.com
ipv4.dns-options:               --
ipv4.dns-priority:              0
ipv4.addresses:                 172.25.250.10/24
ipv4.gateway:                   172.25.250.254
...output omitted...
ipv6.method:                    auto
ipv6.dns:                       --
ipv6.dns-search:                --
ipv6.dns-options:               --
ipv6.dns-priority:              0
ipv6.addresses:                 --
ipv6.gateway:                   --
ipv6.routes:                     --
...output omitted...
GENERAL.NAME:                  Wired connection 1
GENERAL.UUID:                  ec3a15fb-2e26-3254-9433-90c66981e924
GENERAL.DEVICES:                eth0
```

```
GENERAL.IP-IFACE:           eth0
GENERAL.STATE:              activated
GENERAL.DEFAULT:            yes
...output omitted...
```

3.3. 장치 상태를 표시합니다.

```
[root@servera ~]# nmcli dev status
DEVICE      TYPE      STATE      CONNECTION
eth0        ethernet  connected  Wired connection 1
lo          loopback  unmanaged  --
```

3.4. **eth0** 장치의 설정을 표시합니다.

```
[root@servera ~]# nmcli dev show eth0
GENERAL.DEVICE:           eth0
GENERAL.TYPE:              ethernet
GENERAL.HWADDR:            52:54:00:00:FA:0A
GENERAL.MTU:                1500
GENERAL.STATE:              100 (connected)
GENERAL.CONNECTION:         Wired connection 1
GENERAL.CON-PATH:           /org/freedesktop/NetworkManager/ActiveConnection/1
WIRED-PROPERTIES.CARRIER:  on
IP4.ADDRESS[1]:             172.25.250.10/24
IP4.GATEWAY:                172.25.250.254
IP4.ROUTE[1]:               dst = 172.25.250.0/24, nh = 0.0.0.0, mt = 100
IP4.ROUTE[2]:               dst = 0.0.0.0/0, nh = 172.25.250.254, mt = 100
IP4.DNS[1]:                  172.25.250.220
IP4.SEARCHES[1]:             lab.example.com
IP4.SEARCHES[2]:             example.com
IP6.ADDRESS[1]:              fe80::c38a:ac39:36a1:a43c/64
IP6.GATEWAY:                 --
IP6.ROUTE[1]:                dst = fe80::/64, nh = ::, mt = 1024
```

- ▶ 4. 활성 연결과 동일한 IPv4 주소, 네트워크 접두사 및 기본 게이트웨이를 사용하여 정적 연결을 생성합니다. 새 연결의 이름을 **static-addr**로 지정합니다.



경고

시스템 액세스가 기본 네트워크 연결을 통해 제공되므로 기본 네트워크 설정 중에 잘못된 값을 설정하면 시스템에 연결하지 못할 수 있습니다. 시스템에 연결할 수 없는 경우 시스템의 그래픽 디스플레이로 사용된 항목 위에 있는 **Reset** 버튼을 사용하고 다시 시도합니다.

```
[root@servera ~]# nmcli con add con-name static-addr \
  iface eth0 type ethernet ipv4.method manual ipv4.dns 172.25.250.220 \
  ipv4.addresses 172.25.250.10/24 ipv4.gateway 172.25.250.254
Connection 'static-addr' (dc519805-48c4-4b31-b9e9-d3631cf9082c) successfully
added.
```

- ▶ 5. 새 연결을 표시하고 활성화합니다.

5.1. 모든 연결을 봅니다.

```
[root@servera ~]# nmcli con show
NAME           UUID
Wired connection 1  ec3a15fb-2e26-3254-9433-90c66981e924  ethernet  eth0
static-addr      dc519805-48c4-4b31-b9e9-d3631cf9082c  ethernet  --
```

5.2. 활성 연결을 봅니다.

```
[root@servera ~]# nmcli con show --active
NAME           UUID
Wired connection 1  ec3a15fb-2e26-3254-9433-90c66981e924  ethernet  eth0
```

5.3. 새 static-addr 연결을 활성화합니다.

```
[root@servera ~]# nmcli con up static-addr
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)
```

5.4. 새 활성 연결을 봅니다.

```
[root@servera ~]# nmcli con show --active
NAME           UUID
static-addr      dc519805-48c4-4b31-b9e9-d3631cf9082c  ethernet  eth0
```

▶ 6. 부팅할 때 시작되지 않도록 이전 연결을 업데이트합니다. 시스템을 재부팅할 때 static-addr 연결이 사용되는지 확인합니다.

6.1. 부팅할 때 자동으로 시작되지 않도록 원래 연결을 비활성화합니다.

```
[root@servera ~]# nmcli con mod "Wired connection 1" \
connection.autoconnect no
```

6.2. 시스템을 재부팅합니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

6.3. servera 시스템에 로그인하고 static-addr 연결이 활성 연결인지 확인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ nmcli con show --active
NAME           UUID
static-addr      dc519805-48c4-4b31-b9e9-d3631cf9082c  ethernet  eth0
```

▶ 7. 새 네트워크 주소를 사용하여 연결을 테스트합니다.

7.1. IP 주소를 확인합니다.

```
[student@servera ~]$ ip -br addr show eth0
eth0           UP      172.25.250.10/24 fe80::eb21:9a:24de:e8fe/64
```

7.2. 기본 게이트웨이를 확인합니다.

```
[student@servera ~]$ ip route
default via 172.25.250.254 dev eth0 proto static metric 100
172.25.250.0/24 dev eth0 proto kernel scope link src 172.25.250.10 metric 100
```

7.3. DNS 주소로 ping을 실행합니다.

```
[student@servera ~]$ ping -c3 172.25.250.220
PING 172.25.250.220 (172.25.250.220) 56(84) bytes of data.
64 bytes from 172.25.250.220: icmp_seq=1 ttl=64 time=0.777 ms
64 bytes from 172.25.250.220: icmp_seq=2 ttl=64 time=0.431 ms
64 bytes from 172.25.250.220: icmp_seq=3 ttl=64 time=0.272 ms

--- 172.25.250.220 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2045ms
rtt min/avg/max/mdev = 0.272/0.493/0.777/0.210 ms
```

7.4. workstation 시스템에 student 사용자로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-configure
```

이것으로 세션을 완료합니다.

네트워크 구성 파일 편집

목표

구성 파일을 편집하여 네트워크 구성을 수정합니다.

연결 구성 파일

Red Hat Enterprise Linux 8부터 네트워크 구성은 `/etc/NetworkManager/system-connections/` 디렉터리에 저장됩니다. 구성 위치는 `ifcfg` 형식 대신 키 파일 형식을 사용합니다. 그러나 이전에 `/etc/sysconfig/network-scripts/`에 저장한 구성은 계속 작동합니다. `/etc/NetworkManager/system-connections/` 디렉터리는 `nmcli con mod name` 명령을 통한 변경 사항을 저장합니다.

키 파일 형식

NetworkManager는 INI 스타일 키 형식을 사용하여 네트워크 연결 프로필을 저장합니다. 키-값 쌍은 구성을 섹션(그룹)으로 저장합니다. 섹션의 각 구성 키/값 쌍은 설정 사양에 표시된 속성 중 하나입니다. 이 구성 파일은 대부분의 설정을 INI 스타일 형식과 동일한 형식으로 저장합니다. 예를 들어 IP 주소를 `192.168.0.1/24`로 쓰면 정수 배열보다 읽기 쉽습니다.

프로필을 관리하는 권장 방법은 `nmcli` 명령을 사용하는 것이지만 사용자가 수동으로 구성 파일을 생성하거나 수정할 수 있습니다. 구성 파일을 편집한 후 `nmcli con reload` 명령을 실행하여 NetworkManager에 변경 사항을 알립니다.

NetworkManager 설정 및 키 파일 형식 파일 비교

<code>nmcli con mod</code>	<code>* .nmconnection 파일</code>	<code>효과</code>
<code>ipv4.method manual</code>	<code>[ipv4]</code> <code>method=manual</code>	IPv4 주소를 정적으로 구성합니다.
<code>ipv4.method auto</code>	<code>[ipv4]</code> <code>method=auto</code>	DHCPv4 서버에서 구성 설정을 찾습니다. DHCPv4의 정보가 있는 경우에만 고정 주소를 표시합니다.
<code>ipv4.addresses 192.0.2.1/24</code>	<code>[ipv4]</code> <code>address1=192.0.2.1/24</code>	고정 IPv4 주소 및 네트워크 접두사를 설정합니다. 연결 주소가 두 개 이상인 경우 <code>address2</code> 키는 두 번째 주소를 정의하고, <code>address3</code> 키는 세 번째 주소를 정의합니다.
<code>ipv4.gateway 192.0.2.254</code>	<code>[ipv4]</code> <code>gateway=192.0.2.254</code>	기본 게이트웨이를 설정합니다.

nmcli con mod	* .nmconnection 파일	효과
ipv4.dns 8.8.8.8	[ipv4] dns=8.8.8.8	이 이름 서버를 사용하도록 /etc/resolv.conf를 수정합니다.
ipv4.dns-search example.com	[ipv4] dns-search=example.com	/etc/resolv.conf 지시문에서 이 도메인을 사용하도록 search를 수정합니다.
ipv4.ignore-auto-dns true	[ipv4] ignore-auto-dns=true	DHCP 서버에서 DNS 서버 정보를 무시합니다.
ipv6.method manual	[ipv6] method=manual	IPv6 주소를 정적으로 구성합니다.
ipv6.method auto	[ipv6] method=auto	라우터 알림의 SLAAC를 사용하여 네트워크 설정을 구성합니다.
ipv6.method dhcp	[ipv6] method=dhcp	SLAAC가 아닌 DHCPv6를 사용하여 네트워크 설정을 구성합니다.
ipv6.addresses 2001:db8::a/64	[ipv6] address1=2001:db8::a/64	고정 IPv6 주소 및 네트워크 접두사를 설정합니다. 두 개 이상의 주소를 연결에 사용하는 경우 address2 키는 두 번째 주소를 정의하고, address3 키는 세 번째 주소를 정의합니다.
ipv6.gateway 2001:db8::1	[ipv6] gateway=2001:db8::1	기본 게이트웨이를 설정합니다.
ipv6.dns fde2:6494:1e09:2::d	[ipv6] dns=fde2:6494:1e09:2::d	이 이름 서버를 사용하도록 /etc/resolv.conf를 수정합니다. IPv4와 같습니다.
ipv6.dns-search example.com	[ipv6] dns-search=example.com	/etc/resolv.conf 지시문에서 이 도메인을 사용하도록 search를 수정합니다.
ipv6.ignore-auto-dns true	[ipv6] ignore-auto-dns=true	DHCP 서버에서 DNS 서버 정보를 무시합니다.
connection.autoconnect yes	[connection] autoconnect=true	부팅 시 이 연결을 자동으로 활성화합니다.
connection.id ens3	[connection] id>Main eth0	이 연결의 이름입니다.

nmcli con mod	* .nmconnection 파일	효과
<code>connection.interface-name ens3</code>	<code>[connection] interface-name=ens3</code>	연결은 이 이름을 사용하여 네트워크 인터페이스에 바인딩됩니다.
<code>802-3-ethernet.mac-address ...</code>	<code>[802-3-ethernet] mac-address=</code>	연결은 이 MAC 주소를 사용하여 네트워크 인터페이스에 바인딩됩니다.

네트워크 구성 설정

연결 구성 파일을 직접 편집하여 네트워크를 구성할 수도 있습니다. 연결 구성 파일은 개별 네트워크 장치에 대한 소프트웨어 인터페이스를 제어합니다. 이 파일의 이름은 일반적으로 `/etc/NetworkManager/system-connections/name.nmconnection`입니다. 여기서 name은 구성 파일이 제어하는 장치 이름 또는 연결을 나타냅니다.

연결 프로필의 목적에 따라 NetworkManager는 다음 디렉터리를 사용하여 구성 파일을 저장합니다.

- `/etc/NetworkManager/system-connections/` 디렉터리는 사용자가 생성하고 편집한 영구 프로필을 저장합니다. NetworkManager는 해당 프로필을 `/etc/NetworkManager/system-connections/` 디렉터리에 자동으로 복사합니다.
- `/run/NetworkManager/system-connections/` 디렉터리는 시스템을 재부팅할 때 자동으로 제거되는 임시 프로필을 저장합니다.
- `/usr/lib/NetworkManager/system-connections/` 디렉터리는 사전 배포된, 변경 불가능한 프로필을 저장합니다. NetworkManager API를 사용하여 해당 프로필을 편집하면 NetworkManager는 이 프로필을 영구 또는 임시 스토리지에 복사합니다.

정적 IPv4 구성을 위한 샘플 구성 파일 내용:

```
[connection]
id=Main eth0
uuid=27afa607-ee36-43f0-b8c3-9d245cdc4bb3
type=802-3-ethernet
autoconnect=true

[ipv4]
method=manual

[802-3-ethernet]
mac-address=00:23:5a:47:1f:71
```

키 파일 형식에 대한 IPv4 구성 옵션

정적	동적	Either(둘 중 하나)
[ipv4] address1=172.25.0.10/24 gateway=172.25.0.254 dns=172.25.254.254	method=auto	[connection] interface-name=ens3 id>Main eth0 autoconnect=true uuid=f3e8(...)ad3e type=etherent

구성 파일을 수정한 후에는 NetworkManager가 구성 변경 사항을 로드할 수 있도록 `nmcli con reload` 명령을 실행합니다. 연결 프로필에서 `autoconnect` 변수가 `false`로 설정된 경우 연결을 수동으로 활성화 합니다.

```
[root@host ~]# nmcli con reload
[root@host ~]# nmcli con up "static-ens3"
```



참조

`nmcli(1)`, `nm-settings(5)`, `nm-settings-keyfile(5)` 도움말 페이지

자세한 내용은
https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/assembly_manually-creating-networkmanager-profiles-in-key-file-format_configuring-and-managing-networking
 에 있는 Manually Creating NetworkManager Profiles in Key File Format를 참조하십시오.

▶ 연습 가이드

네트워크 구성 파일 편집

이 연습에서는 네트워크 구성 파일을 수동으로 수정하고 새 설정이 적용되도록 합니다.

결과

- 각 시스템에서 추가 네트워크 주소를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-edit
```

지침

- ▶ 1. **workstation** 시스템에서 **ssh** 명령을 사용하여 **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **ip link** 명령을 사용하여 네트워크 인터페이스 이름을 찾습니다.



중요

네트워크 인터페이스 이름은 부팅하는 동안 버스 유형 및 장치 탐지 순서에 따라 결정됩니다.
네트워크 인터페이스 이름은 사용 중인 과정 플랫폼 및 하드웨어에 따라 다를 수 있습니다.

시스템의 이더넷 주소와 연결된 네트워크 인터페이스 이름을 찾습니다. 이 이름을 기록하거나 기억하고 이를 사용하여 후속 명령에서 **enX** 자리 표시자를 대체합니다. 활성 연결 이름은 **Wired connection 1**이고, 구성은 **/etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"** 파일에 있습니다.

```
[student@servera ~]$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
    group default qlen 1000
        link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
    DEFAULT group default qlen 1000
        link/ether 52:54:00:00:fa:0a brd ff:ff:ff:ff:ff:ff
        altname enp0s3
```

```

altname ens3
[student@servera ~]$ nmcli con show --active
NAME           UUID
Wired connection 1  a98933fa-25c0-36a2-b3cd-c056f41758fe
[student@servera ~]$ ls /etc/NetworkManager/system-connections/
'Wired connection 1.nmconnection'

```

- ▶ 3. servera 시스템에서 root 사용자로 전환한 다음, `/etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"` 파일을 편집하여 `10.0.1.1/24` 주소를 추가합니다.

- 3.1. `sudo -i` 명령을 사용하여 root 사용자로 전환합니다.

```

[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#

```

- 3.2. 구성 파일을 편집합니다. 파일의 첫 번째 주소 아래에 `10.0.1.1/24` 주소를 두 번째 주소로 추가합니다.

```

[root@servera ~]# vim /etc/NetworkManager/system-connections/"Wired connection
1.nmconnection"
...output omitted...
[ipv4]
address1=172.25.250.10/24,172.25.250.254
address2=10.0.1.1/24
...output omitted...

```

- ▶ 4. `nmcli` 명령을 사용하여 새 네트워크 주소를 활성화합니다.

- 4.1. NetworkManager의 구성 변경 사항을 다시 로드하여 변경 사항을 읽습니다.

```
[root@servera ~]# nmcli con reload
```

- 4.2. 변경 사항을 사용하여 연결을 활성화합니다.

```

[root@servera ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/2)

```

- ▶ 5. 새 IP 주소가 성공적으로 할당되었는지 확인합니다.

```

[root@servera ~]# ip -br addr show eth0
eth0:      UP      172.25.250.10/24 10.0.1.1/24 fe80::6fed:5a11:4ad4:1bcf/64

```

- ▶ 6. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

- ▶ 7. **serverb** 시스템에서 `/etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"` 파일을 편집하여 **10.0.1.2/24** 주소를 추가하고 새 구성으로드합니다.

7.1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

7.2. 구성 파일을 편집합니다. 파일의 첫 번째 주소 아래에 **10.0.1.2/24** 주소를 두 번째 주소로 추가합니다.

```
[root@serverb ~]# vim /etc/NetworkManager/system-connections/"Wired connection 1.nmconnection"
address1=172.25.250.11/24,172.25.250.254
address2=10.0.1.2/24
```

7.3. NetworkManager의 구성 변경 사항을 다시 로드하여 변경 사항을 읽습니다.

```
[root@serverb ~]# nmcli con reload
```

7.4. 변경 사항을 사용하여 연결을 활성화합니다.

```
[root@serverb ~]# nmcli con up "Wired connection 1"
Connection successfully activated (D-Bus active path: /org/freedesktop/NetworkManager/ActiveConnection/2)
```

7.5. 새 IP 주소가 성공적으로 할당되었는지 확인합니다.

```
[root@serverb ~]# ip -br addr show eth0
eth0      UP      172.25.250.11/24 10.0.1.2/24 fe80::6be8:6651:4280:892c/64
```

- ▶ 8. 새 네트워크 주소를 사용하여 **servera** 및 **serverb** 시스템 간 연결을 테스트합니다.

8.1. **serverb** 시스템에서 **servera** 시스템의 새 주소를 ping합니다.

```
[root@serverb ~]# ping -c3 10.0.1.1
PING 10.0.1.1 (10.0.1.1) 56(84) bytes of data.
64 bytes from 10.0.1.1: icmp_seq=1 ttl=64 time=1.30 ms
```

```
64 bytes from 10.0.1.1: icmp_seq=2 ttl=64 time=0.983 ms
64 bytes from 10.0.1.1: icmp_seq=3 ttl=64 time=0.312 ms

--- 10.0.1.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.312/0.864/1.297/0.410 ms
```

8.2. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

8.3. **student** 사용자로 **servera** 시스템에 액세스하여 **serverb** 시스템의 새 주소를 ping합니다.

```
[student@workstation ~]$ ssh student@servera ping -c3 10.0.1.2
PING 10.0.1.2 (10.0.1.2) 56(84) bytes of data.
64 bytes from 10.0.1.2: icmp_seq=1 ttl=64 time=0.876 ms
64 bytes from 10.0.1.2: icmp_seq=2 ttl=64 time=0.310 ms
64 bytes from 10.0.1.2: icmp_seq=3 ttl=64 time=0.289 ms

--- 10.0.1.2 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.289/0.491/0.876/0.271 ms
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-edit
```

이것으로 섹션을 완료합니다.

호스트 이름 및 이름 확인 구성

목표

서버의 정적 호스트 이름과 이름 확인을 구성하고 결과를 테스트합니다.

시스템 호스트 이름 업데이트

hostname 명령은 정규화된 시스템 호스트 이름을 표시하거나 임시로 수정합니다.

```
[root@host ~]# hostname
host.example.com
```

/etc/hostname 파일에서 정적 호스트 이름을 지정합니다. **hostnamectl** 명령을 사용하여 이 파일을 수정하고 시스템의 정규화된 호스트 이름을 봅니다. 이 파일이 없는 경우 인터페이스에 IP 주소를 할당할 때 역방향 DNS 쿼리를 통해 호스트 이름이 설정됩니다.

```
[root@host ~]# hostnamectl hostname host.example.com
[root@host ~]# hostnamectl status
  Static hostname: host.example.com
    Icon name: computer-vm
      Chassis: vm #
  Machine ID: ace63d6701c2489ab9c0960c0f1afe1d
      Boot ID: 0edf5ba1830c48adbd6babfa08f0b867
  Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
      CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
        Kernel: Linux 5.14.0-70.13.1.el9_0.x86_64
      Architecture: x86-64
  Hardware Vendor: Red Hat
  Hardware Model: OpenStack Compute
[root@host ~]# cat /etc/hostname
host.example.com
```



중요

Red Hat Enterprise Linux 7 이상 버전에서 정적 호스트 이름은 **/etc/hostname** 파일에 저장됩니다. Red Hat Enterprise Linux 6 및 이전 버전에서는 호스트 이름을 **/etc/sysconfig/network** 파일에 변수로 저장합니다.

이름 확인 구성

stub resolver는 호스트 이름을 IP 주소로 변환하거나 그 반대로 변환합니다. **/etc/nsswitch.conf** 파일의 구성에 따라 찾아볼 위치를 결정합니다. 기본적으로 먼저 **/etc/hosts** 파일을 사용하여 쿼리를 해결하려고 합니다.

```
[root@host ~]# cat /etc/hosts
127.0.0.1      localhost localhost.localdomain localhost4 localhost4.localdomain4
::1            localhost localhost.localdomain localhost6 localhost6.localdomain6
172.25.254.254 classroom.example.com
172.25.254.254 content.example.com
```

`getent hosts hostname` 명령은 `/etc/hosts` 파일을 사용하여 호스트 이름 확인을 테스트합니다. `/etc/hosts` 파일에서 항목을 찾을 수 없는 경우 기본적으로 stub resolver는 DNS 이름 서버를 사용하여 호스트 이름을 조회합니다. `/etc/resolv.conf` 파일은 이 쿼리를 수행하는 방법을 제어합니다.

- **search**: 짧은 호스트 이름으로 시도할 도메인 이름 목록입니다. 동일한 파일에서 **search** 또는 **domain**을 설정해야 합니다. 둘 다 설정하면 마지막 항목만 적용됩니다. 자세한 내용은 `resolv.conf(5)`를 참조하십시오.
- **nameserver** : 쿼리할 이름 서버의 IP 주소입니다. 이름 서버 하나가 다운된 경우 백업을 제공하기 위해 최대 3개의 이름 서버 지시문을 지정할 수 있습니다.

```
[root@host ~]# cat /etc/resolv.conf
# Generated by NetworkManager
domain example.com
search example.com
nameserver 172.25.254.254
```

NetworkManager는 연결 구성 파일의 DNS 설정을 사용하여 `/etc/resolv.conf` 파일을 업데이트합니다. 연결을 수정하려면 `nmcli` 명령을 사용합니다.

```
[root@host ~]# nmcli con mod ID ipv4.dns IP
[root@host ~]# nmcli con down ID
[root@host ~]# nmcli con up ID
[root@host ~]# cat /etc/NetworkManager/system-connections/ID
...output omitted...
[ipv4]
...output omitted...
dns=8.8.8.8;
...output omitted...
```

`nmcli con mod ID ipv4.dns IP` 명령의 기본 동작은 이전 DNS 설정을 제공된 새 IP 목록으로 교체하는 것입니다. `nmcli` 명령의 `ipv4.dns` 옵션 앞에 있는 더하기(+) 또는 빼기(-) 문자는 각각 개별 항목을 추가하거나 제거합니다.

```
[root@host ~]# nmcli con mod ID +ipv4.dns IP
```

다음 예시에서 `static-ens3` 연결의 이름 서버 목록에 IPv6 IP 주소가 `2001:4860:4860::8888`인 DNS 서버를 추가하려면 다음을 실행합니다.

```
[root@host ~]# nmcli con mod static-ens3 +ipv6.dns 2001:4860:4860::8888
```

**참고**

정적 IPv4 및 IPv6 DNS 설정은 `/etc/resolv.conf`에서 `nameserver` 지시문이 됩니다.
듀얼 스택 시스템에서는 스택 중 하나에서 네트워킹 문제가 발생할 때를 대비하여 IPv4로 연결할 수 있는 이름 서버 1개 이상과 IPv6 이름 서버 1개가 표시되도록 합니다(듀얼 스택 시스템이라고 가정).

DNS 이름 확인 테스트

`host HOSTNAME` 명령은 DNS 서버 연결을 테스트할 수 있습니다.

```
[root@host ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
[root@host ~]# host 172.25.254.254
254.254.25.172.in-addr.arpa domain name pointer classroom.example.com.
```

**중요**

DHCP에서는 `/etc/NetworkManager/system-connections/` 디렉터리의 관련 인터페이스 구성 파일에 `ignore-auto-dns = yes`를 지정하지 않는 한, 인터페이스가 시작될 때 `/etc/resolv.conf` 파일을 자동으로 다시 작성합니다.

`nmcli` 명령을 사용하여 이 항목을 설정합니다.

```
[root@host ~]# nmcli con mod "static-ens3" ipv4.ignore-auto-dns yes
```

`dig HOSTNAME` 명령을 사용하여 DNS 서버 연결을 테스트합니다.

```
[root@host ~]# dig classroom.example.com

; <>> DiG 9.16.23-RH <>> classroom.example.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 3451
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 13, ADDITIONAL: 27

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
; COOKIE: 947ea2a936353423c3bc0d5f627cc1ae7147460e10d2777c (good)
;; QUESTION SECTION:
;classroom.example.com. IN A

;; ANSWER SECTION:
classroom.example.com. 85326 IN A 172.25.254.254
...output omitted...
```

`host` 및 `dig` 명령은 둘 다 `/etc/hosts` 파일의 구성을 표시하지 않습니다. `/etc/hosts` 파일을 테스트 하려면 `getent hosts HOSTNAME` 명령을 사용합니다.

```
[root@host ~]# getent hosts classroom.example.com  
172.25.254.254 classroom.example.com
```



참조

`nmcli(1)`, `hostnamectl(1)`, `hosts(5)`, `getent(1)`, `host(1)`, `dig(1)`, `getent(1)`, `resolv.conf(5)` 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index

에 있는 Configuring and Managing Networking Guide를 참조하십시오.

▶ 연습 가이드

호스트 이름 및 이름 확인 구성

이 연습에서는 시스템의 정적 호스트 이름, `/etc/hosts` 파일, DNS 이름 확인자를 수동으로 구성합니다.

결과

- 사용자 지정 호스트 이름을 설정합니다.
- 이름 확인 설정을 구성합니다.

시작하기 전에

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-hostnames
```

지침

- ▶ 1. `student` 사용자로 `servera`에 로그인한 후 `root` 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@testa ~]$ sudo -i
[sudo] password for student: student
[root@testa ~]#
```

- ▶ 2. 현재 호스트 이름 설정을 봅니다.

- 2.1. 현재 호스트 이름을 표시합니다.

```
[root@testa ~]# hostname
testa
```

- 2.2. 호스트 이름 상태를 표시합니다. 로컬로 구성된 영구적인 호스트 이름이 `Static hostname` 필드에 표시됩니다. DHCP 또는 DNS 네트워크 서비스에서 가져온 현재 런타임 호스트 이름이 `Transient hostname` 필드에 표시됩니다.

```
[root@testa ~]# hostnamectl status
  Static hostname: servera.lab.example.com
  Transient hostname: testa
    Icon name: computer-vm
      Chassis: vm #
  Machine ID: ace63d6701c2489ab9c0960c0f1afe1d
```

```

Boot ID: 03bf1d5518bd43b4a25cfe9a18d5a46a
Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
    CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
        Kernel: Linux 5.14.0-70.13.1.el9_0.x86_64
    Architecture: x86-64
Hardware Vendor: Red Hat
    Hardware Model: OpenStack Compute

```

▶ 3. 현재 정적 호스트 이름과 일치하도록 정적 호스트 이름을 설정합니다.

- 3.1. 호스트 이름 및 호스트 이름 구성 파일을 변경합니다.

```
[root@testa ~]# hostnamectl hostname \
servera.lab.example.com
```

- 3.2. 네트워크를 시작할 때 호스트 이름을 제공하는 **/etc/hostname** 파일 내용을 봅니다.

```
[root@testa ~]# cat /etc/hostname
servera.lab.example.com
```

- 3.3. 로그아웃한 다음 **student** 사용자로 **servera**에 로그인합니다. **root** 사용자로 전환하여 업데이트된 호스트 이름을 표시하도록 명령 프롬프트를 변경합니다.

```

[root@testa ~]# exit
logout
[student@testa ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student:
[root@servera ~]#

```

- 3.4. 호스트 이름 상태를 표시합니다. 이제 정적 호스트 이름이 구성되었으므로 임시 호스트 이름은 표시되지 않습니다.

```

[root@servera ~]# hostnamectl status
Static hostname: servera.lab.example.com
    Icon name: computer-vm
    Chassis: vm
    Machine ID: 63b272eae8d5443ca7aaa5593479b25f
    Boot ID: ef299e0e957041ee81d0617fc98ce5ef
Virtualization: kvm
Operating System: Red Hat Enterprise Linux 9.0 (Plow)
    CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
        Kernel: Linux 5.14.0-70.el9.x86_64
    Architecture: x86-64
Hardware Vendor: Red Hat
    Hardware Model: OpenStack Compute

```

▶ 4. 임시로 호스트 이름을 **testname**(으)로 변경합니다.

4.1. 호스트 이름을 변경합니다.

```
[root@servera ~]# hostname testname
```

4.2. 현재 호스트 이름을 표시합니다.

```
[root@servera ~]# hostname
testname
```

4.3. 네트워크를 시작할 때 호스트 이름을 제공하는 **/etc/hostname** 파일 내용을 봅니다.

```
[root@servera ~]# cat /etc/hostname
servera.lab.example.com
```

4.4. 시스템을 재부팅합니다.

```
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

4.5. **servera**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

4.6. 현재 호스트 이름을 표시합니다.

```
[root@servera ~]# hostname
servera.lab.example.com
```

▶ 5. 강의실 서버의 로컬 별칭으로 **class**를 추가하고 해당 별칭으로 서버를 ping 할 수 있는지 확인합니다.

5.1. **classroom.example.com** 서버의 IP 주소를 조회합니다.

```
[root@servera ~]# host classroom.example.com
classroom.example.com has address 172.25.254.254
```

5.2. **/etc/hosts** 파일을 업데이트하여 **172.25.254.254** IP 주소에 액세스하도록 **class** 서버를 추가합니다. 다음 예제에서는 예상되는 **/etc/hosts** 파일 내용을 보여줍니다.

```
[root@servera ~]# vim /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
172.25.254.254 classroom.example.com classroom class
```

5.3. **class** 서버의 IP 주소를 조회합니다.

```
[root@servera ~]# host class
Host class not found: 3(NXDOMAIN)
[root@servera ~]# getent hosts class
172.25.254.254 classroom.example.com classroom class
```

5.4. **ping** 명령을 사용하여 **class** 서버에 패킷을 보냅니다.

```
[root@servera ~]# ping -c3 class
PING classroom.example.com (172.25.254.254) 56(84) bytes of data.
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=1 ttl=63 time=1.21
ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=2 ttl=63 time=0.688
ms
64 bytes from classroom.example.com (172.25.254.254): icmp_seq=3 ttl=63 time=0.559
ms

--- classroom.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2046ms
rtt min/avg/max/mdev = 0.559/0.820/1.214/0.283 ms
```

5.5. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-hostnames
```

이것으로 섹션을 완료합니다.

▶ 랩

네트워킹 관리

이 랩에서는 Red Hat Enterprise Linux 서버의 네트워킹 설정을 구성합니다.

결과

- 기본 네트워크 인터페이스에 대해 고정 IPv4 주소 2개를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-review
```

지침

- student** 사용자로 **serverb** 시스템에 로그인합니다. **root** 사용자로 전환합니다.
- 표의 설정을 사용하여 정적 네트워크 구성으로 연결을 생성합니다.

매개 변수	설정
연결 이름	랩
인터페이스 이름	enX(다양할 수 있으며 MAC 주소가 52:54:00:00:fa:0b 인 인터페이스 사용)
IP 주소	172.25.250.11/24
게이트웨이 주소	172.25.250.254
DNS 주소	172.25.250.254

- 자동으로 시작되도록 새 연결을 구성합니다. 다른 연결은 자동으로 시작되면 안 됩니다.
- 10.0.1.1/24** IP 주소를 사용하도록 새 연결을 수정합니다.
- private** 이름으로 **10.0.1.1** IP 주소를 참조할 수 있도록 **hosts** 파일을 구성합니다.
- 시스템을 재부팅합니다.
- serverb** 시스템이 초기화되었는지 확인합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade net-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

네트워킹 관리

이 랩에서는 Red Hat Enterprise Linux 서버의 네트워킹 설정을 구성합니다.

결과

- 기본 네트워크 인터페이스에 대해 고정 IPv4 주소 2개를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start net-review
```

지침

- student** 사용자로 **serverb** 시스템에 로그인합니다. **root** 사용자로 전환합니다.

- 1.1. **serverb** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

2. 표의 설정을 사용하여 정적 네트워크 구성으로 연결을 생성합니다.

매개 변수	설정
연결 이름	랩
인터페이스 이름	enX(다양할 수 있으며 MAC 주소가 52:54:00:00:fa:0b 인 인터페이스 사용)
IP 주소	172.25.250.11/24
게이트웨이 주소	172.25.250.254
DNS 주소	172.25.250.254

- 2.1. 인터페이스 이름과 현재 활성 연결의 이름을 결정합니다.



중요

연결 이름은 사용 중인 교육 과정 플랫폼 및 하드웨어에 따라 다를 수 있습니다. 이 솔루션은 인터페이스 이름이 **eth0**이며 연결 이름은 **System eth0**이라고 가정합니다. 표에서 인터페이스와 연결된 현재 연결 이름(예: **System eth0** 또는 **Wired connection 1**)을 찾습니다. 찾은 연결 이름을 사용하여 이 연습 전반에서 **System eth0**을 대체합니다(다른 경우).

```
[root@serverb ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:00:fa:0b brd ff:ff:ff:ff:ff:ff
        altname enp0s3
        altname ens3
[root@serverb ~]# nmcli con show --active
NAME           UUID             TYPE      DEVICE
System eth0   5fb06bd0-0bb0-7ffb-45f1-d6edd65f3e03  ethernet  eth0
```

2. 지침에 있는 표 정보에 따라 **lab** 연결 프로필을 생성합니다. 이전 **ip link** 명령의 출력에 표시된 네트워크 인터페이스와 프로필을 연결합니다.

```
[root@serverb ~]# nmcli con add con-name lab iface eth0 type ethernet \
    ipv4.method manual ipv4.dns 172.25.250.254 \
    ipv4.addresses 172.25.250.11/24 ipv4.gateway 172.25.250.254
Connection 'lab' (d5b4815d-231b-43c0-8565-445e3a07b97a) successfully added.
```

3. 자동으로 시작되도록 새 연결을 구성합니다. 다른 연결은 자동으로 시작되면 안 됩니다.

```
[root@serverb ~]# nmcli con mod "lab" connection.autoconnect yes
[root@serverb ~]# nmcli con mod "System eth0" connection.autoconnect no
```

4. **10.0.1.1/24** IP 주소를 사용하도록 새 연결을 수정합니다.

```
[root@serverb ~]# nmcli con mod "lab" +ipv4.addresses 10.0.1.1/24
```

또는 구성 파일을 편집하여 **10.0.1.1/24** 주소를 두 번째 주소로 추가합니다.

```
[root@serverb ~]# vim /etc/NetworkManager/system-connections/lab.nmconnection
address2=10.0.1.1/24
```

5. **private** 이름으로 **10.0.1.1** IP 주소를 참조할 수 있도록 **hosts** 파일을 구성합니다.

```
[root@serverb ~]# echo "10.0.1.1 private" >> /etc/hosts
```

6. 시스템을 재부팅합니다.

```
[root@serverb ~]# systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

7. **serverb** 시스템이 초기화되었는지 확인합니다.

```
[student@workstation ~]$ ping -c3 serverb
PING serverb.lab.example.com (172.25.250.11) 56(84) bytes of data.
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=1 ttl=64
time=0.478 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=2 ttl=64
time=0.504 ms
64 bytes from serverb.lab.example.com (172.25.250.11): icmp_seq=3 ttl=64
time=0.513 ms
--- serverb.lab.example.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 78ms
rtt min/avg/max/mdev = 0.478/0.498/0.513/0.023 ms
```

평가

workstation 시스템에서 **student** 사용자로 **Lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade net-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish net-review
```

이것으로 섹션을 완료합니다.

요약

- **NetworkManager** 데몬은 네트워크 구성을 모니터링하고 관리합니다.
- **NetworkManager** 데몬을 사용하여 네트워크 설정을 구성하기 위한 **nmcli** 명령줄 터입니다.
- Red Hat Enterprise Linux 9부터 네트워크 구성의 기본 위치는 **/etc/NetworkManager/system-connections** 디렉터리입니다.
- 시스템의 정적 호스트 이름은 **/etc/hostname** 파일에 저장됩니다.
- **hostnamectl** 명령은 시스템의 호스트 이름 및 관련 설정의 상태를 수정하거나 표시합니다.

네트워크 연결 스토리지 액세스

목적

NFS 프로토콜을 사용하여 네트워크 연결 스토리지에 액세스합니다.

목표

- NFS 내보내기 정보를 식별하고, 마운트 지점으로 사용할 디렉터리를 생성하고, **mount** 명령을 사용하거나 **/etc/fstab** 파일을 구성하여 NFS 내보내기를 마운트하고, **umount** 명령을 사용하여 NFS 내보내기를 마운트 해제합니다.
- 자동 마운터 사용의 이점을 설명하고 직접 및 간접 맵을 사용하여 NFS 내보내기를 자동으로 마운트합니다.

섹션

- NFS를 사용하여 네트워크 연결 스토리지 관리(안내에 따른 연습)
- 네트워크 연결 스토리지 자동 마운트(안내에 따른 연습)

랩

네트워크 연결 스토리지 액세스

NFS를 사용한 네트워크 연결 스토리지 관리

목표

NFS 내보내기 정보를 식별하고, 마운트 지점으로 사용할 디렉터리를 생성하고, **mount** 명령을 사용하거나 /etc/fstab 파일을 구성하여 NFS 내보내기를 마운트하고, **umount** 명령을 사용하여 NFS 내보내기를 마운트 해제합니다.

내보낸 NFS 디렉터리 액세스

NFS(네트워크 파일 시스템)는 Linux, Unix 및 유사 운영 체제에서 기본 네트워크 파일 시스템으로 사용하는 인터넷 표준 프로토콜입니다. NFS는 기본 Linux 권한 및 파일 시스템 특성을 지원하는 개방형 표준입니다.

기본적으로 Red Hat Enterprise Linux 9에서는 NFS 버전 4.2를 사용합니다. RHEL은 NFSv3 및 NFSv4 프로토콜을 둘 다 지원합니다. NFSv3는 TCP 또는 UDP 전송 프로토콜을 사용할 수 있지만 NFSv4는 TCP 연결만 허용합니다.

NFS 서버는 디렉터리를 내보냅니다. NFS 클라이언트는 내보낸 디렉터리를 기존 로컬 마운트 지점 디렉터리에 마운트합니다. NFS 클라이언트는 내보낸 디렉터리를 다양한 방법으로 마운트할 수 있습니다.

- **mount** 명령을 사용하여 **수동으로**
- **/etc/fstab** 파일의 항목을 구성하여 **부팅 시 영구적으로**
- 자동 마운터 메서드를 구성하여 **온디맨드로**

autofs 서비스 및 **systemd.automount** 기능을 포함하는 자동 마운터 메서드는 **Automount Network-Attached Storage** 섹션에 설명되어 있습니다. 내보낸 NFS 디렉터리를 가져오려면 **nfs-utils** 패키지를 설치하여 수동 마운트 또는 자동 마운트에 필요한 클라이언트 툴을 가져와야 합니다.

```
[root@host ~]# dnf install nfs-utils
```

RHEL에서는 NFS 프로토콜과 동일한 메서드를 사용하거나 SMB(Server Message Block) 또는 CIFS(Common Internet File System) 프로토콜을 사용하여 Microsoft Windows 시스템의 공유 디렉터리를 마운트할 수도 있습니다. 마운트 옵션은 프로토콜별로 다르며 Windows Server 또는 Samba Server 구성에 따라 다릅니다.

서버의 내보낸 NFS 디렉터리 쿼리

NFS 프로토콜은 NFSv3와 NFSv4 사이에 크게 변경되었습니다. 사용 가능한 내보내기를 보기 위해 서버를 쿼리하는 메서드는 프로토콜 버전마다 다릅니다.

NFSv3에서는 RPC 프로토콜을 사용했는데, 이 프로토콜에는 **rpcbind** 서비스를 실행하기 위해 NFSv3 연결을 지원하는 파일 서버가 필요합니다. NFSv3 클라이언트는 서버의 포트 111에서 **rpcbind** 서비스에 연결하여 NFS 서비스를 요청합니다. 서버는 NFS 서비스의 현재 포트를 사용하여 응답합니다. RPC 기반 NFSv3 서버에서 사용 가능한 내보내기를 쿼리하려면 **showmount** 명령을 사용합니다.

```
[root@host ~]# showmount --exports server
Export list for server
/shares/test1
/shares/test2
```

NFSv4 프로토콜에서는 NFS 트랜잭션에 레거시 RPC 프로토콜을 사용하지 않습니다. NFSv4만 지원하는 서버에서 **showmount** 명령을 사용하면 응답을 수신하지 않은 채 시간이 초과됩니다. 서버에서 **rpcbind** 서비스가 실행되고 있지 않기 때문입니다. 그러나 NFSv4 서버를 쿼리하는 것이 NFSv3 서버를 쿼리하는 것보다 더 간단합니다.

NFSv4에서는 서버의 내보낸 디렉터리에 대한 모든 경로가 포함된 내보내기 트리가 도입되었습니다. 내보낸 디렉터리를 모두 보려면 서버 내보내기 트리의 루트(/)를 마운트합니다. 내보내기 트리의 루트를 마운트하면 모든 내보낸 디렉터리의 탐색 가능한 경로가 트리의 루트 디렉터리 하위로 제공되지만, 내보낸 디렉터리는 마운트('바인딩')되지 않습니다.

```
[root@host ~]# mkdir /mountpoint
[root@host ~]# mount server:/ /mountpoint
[root@host ~]# ls /mountpoint
```

마운트된 내보내기 트리를 검색하는 동안 NFSv4 내보내기를 마운트하려면 디렉터리를 내보낸 디렉터리 경로로 변경하십시오. 또는 내보낸 단일 디렉터리를 마운트하려면 **mount** 명령을 내보낸 단일 디렉터리의 전체 경로 이름과 함께 사용합니다. Kerberos 보안을 사용하는 내보낸 디렉터리는 내보내기 경로 이름은 볼 수 있지만 내보내기 트리를 검색하는 동안 디렉터리를 마운트하거나 디렉터리에 액세스할 수는 없습니다. Kerberos로 보호된 공유를 마운트하려면 서버를 추가로 구성하고 Kerberos 사용자 자격 증명을 사용해야 합니다. 해당 내용은 Red Hat Security: Identity Management and Active Directory Integration(RH362) 교육 과정에서 설명합니다.

내보낸 NFS 디렉터리 수동 마운트

마운트할 NFS 내보내기를 식별한 후 로컬 마운트 지점이 아직 없는 경우 이를 생성합니다. **/mnt** 디렉터리는 임시 마운트 지점으로 사용할 수 있지만 장기 또는 영구 마운트에는 **/mnt**를 사용하지 않는 것이 좋습니다.

```
[root@host ~]# mkdir /mountpoint
```

로컬 볼륨 파일 시스템과 마찬가지로 NFS 내보내기를 마운트하여 해당 콘텐츠에 액세스합니다. NFS 공유는 권한 있는 사용자만 일시적으로 또는 영구적으로 마운트할 수 있습니다.

```
[root@host ~]# mount -t nfs -o rw, sync server:/export /mountpoint
```

-t nfs 옵션은 NFS 파일 시스템 유형을 지정합니다. 그러나 **mount** 명령이 **server:/export** 구문을 탐지하는 경우 명령의 기본값은 NFS 유형입니다. **-o** 플래그를 사용하면 쉼표로 구분된 옵션 목록을 **mount** 명령에 추가할 수 있습니다. 예에서 **rw** 옵션은 내보낸 파일 시스템이 읽기/쓰기 액세스를 통해 마운트되도록 지정합니다. **sync** 옵션은 내보낸 파일 시스템에 대한 동기 트랜잭션을 지정합니다. 이 방법은 트랜잭션을 완료하거나 실패로 반환해야 하는 모든 프로덕션 네트워크 마운트에 사용하는 것이 좋습니다.

수동 **mount** 명령을 사용하는 것은 영구적이지 않습니다. 시스템을 재부팅하면 해당 NFS 내보내기가 더 이상 마운트되지 않습니다. 수동 마운트는 내보낸 디렉터리에 대한 임시 액세스를 제공하거나 NFS 내보내기를 영구적으로 마운트하기 전에 테스트 마운트하는 데 유용합니다.

내보낸 NFS 디렉터리 영구 마운트

NFS 내보내기를 영구적으로 마운트하려면 **/etc/fstab** 파일을 편집하여 수동 마운트와 구문이 유사한 마운트 항목을 추가합니다.

```
[root@host ~]# vim /etc/fstab
...
server:/export  /mountpoint  nfs  rw  0  0
```

그리면 마운트 지점만 사용하여 NFS 내보내기를 마운트할 수 있습니다. **mount** 명령은 **/etc/fstab** 파일의 일치하는 항목에서 NFS 서버 및 마운트 옵션을 가져옵니다.

```
[root@host ~]# mount /mountpoint
```

내보낸 NFS 디렉터리 마운트 해제

권한 있는 사용자로 **umount** 명령을 사용하여 NFS 내보내기를 마운트 해제합니다. **/etc/fstab** 파일이 존재하는 경우 공유를 마운트 해제해도 파일에서 해당 항목이 제거되지 않습니다. **/etc/fstab** 파일의 항목은 영구적이며 부팅 중 다시 마운트됩니다.

```
[root@host ~]# umount /mountpoint
```

마운트된 디렉터리는 간혹 마운트 해제할 수 없으며 장치가 사용 중이라는 오류를 반환합니다. 애플리케이션이 파일 시스템 내에서 파일을 열어 두었거나 일부 사용자의 쉘에서 작업 중인 디렉터리가 마운트된 파일 시스템의 루트 디렉터리 또는 그 아래에 있어 장치가 사용 중입니다.

오류를 해결하려면 활성 쉘 창을 확인하고 **cd** 명령을 사용하여 마운트된 파일 시스템을 종료합니다. 이후 파일 시스템을 마운트 해제하려는 시도가 계속 실패하면 **lsof**(list open files) 명령을 사용하여 마운트 지점을 쿼리합니다. **lsof** 명령은 열려 있는 파일 이름 목록과 파일을 열린 상태로 유지하는 프로세스를 반환합니다.

```
[root@host ~]# lsof /mountpoint
COMMAND  PID  USER   FD   TYPE   DEVICE SIZE/OFF NODE NAME
program  5534 user    txt   REG  252.4  910704    128 /home/user/program
```

이 정보를 사용하여 이 파일 시스템에서 파일을 사용 중인 모든 프로세스를 정상적으로 종료하고 마운트 해제를 다시 시도합니다. 중요한 시나리오에 한해 애플리케이션을 정상적으로 종료할 수 없는 경우 프로세스를 종료하여 파일을 종료합니다. 또는 **umount -f** 옵션을 사용하여 강제로 마운트 해제합니다. 이 경우 열려 있는 모든 파일의 작성되지 않은 데이터가 손실될 수 있습니다.



참조

mount(8), **umount(8)**, **showmount(8)**, **fstab(5)**, **mount.nfs(8)**, **nfsconf(8)**,
rpcbind(8) 도움말 페이지

▶ 연습 가이드

NFS를 사용한 네트워크 연결 스토리지 관리

이 연습에서는 `/etc/fstab` 파일을 수정하여 부팅 시 NFS 내보내기를 영구적으로 마운트합니다.

결과

- `mount` 명령으로 NFS 서버를 테스트합니다.
- 시스템 재부팅 후에도 변경 사항이 저장되도록 `/etc/fstab` 구성 파일에 NFS 내보내기를 구성합니다.

시작하기 전에

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start netstorage-nfs
```

지침

운송 회사는 중앙 NFS 서버인 `serverb`를 사용하여 다양한 내보내기 문서 및 디렉터리를 호스팅합니다. `servera`의 사용자는 모두 `admin` 그룹의 멤버이며, 영구적으로 마운트된 NFS 내보내기에 대한 액세스 권한이 있어야 합니다.

다음 목록은 이 연습을 완료하기 위한 환경 특성을 제공합니다.

- `serverb` 시스템은 몇 가지 텍스트 파일이 포함된 `/shares/public` 디렉터리를 내보냅니다.
- `admin` 그룹(`admin1, sysmanager1`) 회원은 내보낸 `/shares/public` 디렉터리에 대한 읽기 및 쓰기 권한이 있습니다.
- `servera`의 마운트 지점은 `/public` 디렉터리여야 합니다.
- 모든 사용자 암호는 `redhat`으로 설정되어 있습니다.
- `nfs-utils` 패키지가 이미 설치되어 있습니다.

▶ 1. `servera`에 `student` 사용자로 로그인한 후 `root` 사용자로 전환합니다.

1.1. `servera`에 `student` 사용자로 로그인한 후 `root` 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

▶ 2. serverb에서 servera를 NFS 클라이언트로 사용하여 NFS 서버를 테스트합니다.

- 2.1. servera 시스템에 /public 마운트 지점을 생성합니다.

```
[root@servera ~]# mkdir /public
```

- 2.2. servera에서 serverb의 /shares/public NFS 내보내기가 /public 디렉터리에 성공적으로 마운트되었는지 확인합니다.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/public /public
```

- 2.3. 마운트된 NFS 내보내기의 콘텐츠를 나열합니다.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r--. 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r--. 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r--. 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r--. 1 root admin 27 Apr  8 22:36 Trackings.txt
```

- 2.4. 마운트된 NFS 내보내기에 대한 mount 명령 옵션을 살펴봅니다.

```
[root@servera ~]# mount | grep public
serverb.lab.example.com:/shares/public on /public type nfs4
(rw,relatime,vers=4.2,rsize=262144,wszie=262144,namlen=255,sync
,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
```

- 2.5. NFS 내보내기를 마운트 해제합니다.

```
[root@servera ~]# umount /public
```

▶ 3. /shares/public 내보내기가 영구적으로 마운트되도록 servera를 구성합니다.

- 3.1. /etc/fstab 파일을 편집합니다.

```
[root@servera ~]# vim /etc/fstab
```

다음 행을 파일의 끝에 추가합니다.

```
serverb.lab.example.com:/shares/public /public nfs rw,sync 0 0
```

- 3.2. 내보낸 디렉터리를 마운트합니다.

```
[root@servera ~]# mount /public
```

- 3.3. 내보낸 디렉터리의 콘텐츠를 나열합니다.

```
[root@servera ~]# ls -l /public
total 16
-rw-r--r-- 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r-- 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r-- 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r-- 1 root admin 27 Apr  8 22:36 Trackings.txt
```

3.4. **servera** 시스템을 재부팅합니다.

```
[root@servera ~]# systemctl reboot
```

- ▶ 4. **servera** 가 재부팅을 완료하면 **servera**에 **admin1** 사용자로 로그인하여 영구적으로 마운트된 NFS 내보내기를 테스트합니다.

4.1. **admin1** 사용자로 **servera**에 로그인합니다.

```
[student@workstation ~]$ ssh admin1@servera
[admin1@servera ~]$
```

4.2. **/public** 디렉터리에 마운트된 NFS 내보내기를 테스트합니다.

```
[admin1@servera ~]$ ls -l /public
total 16
-rw-r--r-- 1 root admin 42 Apr  8 22:36 Delivered.txt
-rw-r--r-- 1 root admin 46 Apr  8 22:36 NOTES.txt
-rw-r--r-- 1 root admin 20 Apr  8 22:36 README.txt
-rw-r--r-- 1 root admin 27 Apr  8 22:36 Trackings.txt
[admin1@servera ~]$ cat /public/NOTES.txt
###In this file you can log all your notes###
[admin1@servera ~]$ echo "This is a test" > /public/Test.txt
[admin1@servera ~]$ cat /public/Test.txt
This is a test
```

4.3. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[admin1@servera ~]$ exit
logout
Connection to servera closed.
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netstorage-nfs
```

이것으로 섹션을 완료합니다.

네트워크 연결 스토리지 자동 마운트

목표

자동 마운터 사용의 이점을 설명하고 직접 및 간접 맵을 사용하여 NFS 내보내기를 자동으로 마운트합니다.

자동 마운터를 사용하여 NFS 내보내기 마운트

자동 마운터는 파일 시스템 및 NFS 내보내기를 온디맨드로 자동 마운트하고, 마운트된 리소스가 현재 더 이상 사용되지 않는 경우 파일 시스템 및 NFS 내보내기를 자동으로 마운트 해제하는 서비스(**autofs**)입니다.

자동 마운터 기능은 권한이 없는 사용자에게 `mount` 명령을 사용할 수 있는 충분한 권한이 없는 문제를 해결하기 위해 고안되었습니다. 일반 사용자가 CD, DVD와 같은 이동식 미디어 및 이동식 디스크 드라이브에 액세스하려면 `mount` 명령을 사용해야 합니다. 또한 부팅 시 `/etc/fstab` 구성을 사용하여 로컬 또는 원격 파일 시스템을 마운트해야 일반 사용자가 마운트 해제된 파일 시스템을 마운트하고 해당 시스템에 액세스할 수 있습니다.

자동 마운터 구성 파일은 `/etc/fstab` 항목과 유사한 방식으로 파일 시스템 마운트 정보로 채워집니다. `/etc/fstab` 파일 시스템은 시스템 부팅 중 마운트되고 시스템을 종료하거나 다른 개입이 있을 때까지 마운트된 상태로 유지되지만, 자동 마운터 파일 시스템은 시스템 부팅 중에 반드시 마운트되는 것은 아닙니다. 대신 자동 마운터 제어 파일 시스템은 사용자 또는 애플리케이션이 파일에 액세스하기 위해 파일 시스템 마운트 지점을 입력할 때 온디맨드로 마운트됩니다.

자동 마운터 이점

파일 시스템은 프로그램에서 열려 있는 파일을 읽고 쓸 때만 리소스를 사용하기 때문에 자동 마운터 파일 시스템의 리소스 사용량은 부팅 시 마운트된 파일 시스템과 동일합니다. 마운트되었지만 유휴 상태인 파일 시스템과 마운트되지 않은 파일 시스템은 동일한 양의 리소스를 사용합니다.

자동 마운터의 이점은 파일 시스템을 더 이상 사용하지 않을 때마다 마운트 해제하여 파일 시스템이 열려 있는 동안 발생하는 예기치 않은 손상으로부터 보호한다는 것입니다. 파일 시스템을 다시 마운트하도록 지정하면 마지막 시스템 부팅 중 몇 달 전에 마운트한 구성을 여전히 사용할 수 있는 `/etc/fstab` 마운트와 달리, `autofs` 서비스는 최신 마운트 구성을 사용합니다. 또한 NFS 서버 구성에 중복 서버 및 경로가 포함된 경우 자동 마운터는 새 파일 시스템이 요청될 때마다 가장 빠른 연결을 선택할 수 있습니다.

자동 마운터 `autofs` 서비스 메서드

`autofs` 서비스는 NFS 및 SMB 파일 공유 프로토콜을 포함하여 `/etc/fstab` 파일과 같은 로컬 및 원격 파일 시스템을 지원하고, 보안 매개 변수를 포함하여 동일한 프로토콜별 마운트 옵션을 지원합니다. 자동 마운터를 통해 마운트한 파일 시스템은 기본적으로 모든 사용자가 사용할 수 있지만 액세스 권한 옵션을 통해 제한할 수 있습니다.

자동 마운터는 표준 `mount` 및 `umount` 명령을 사용하여 파일 시스템을 관리하는 클라이언트 측 구성이므로 사용 중인 자동 마운트 파일 시스템은 `/etc/fstab`를 사용하여 마운트한 파일 시스템과 동일하게 작동합니다. 차이점은 마운트 지점에 액세스할 때까지 자동 마운터 파일 시스템이 마운트 해제된 상태로 유지되어 파일 시스템이 즉시 마운트되고 파일 시스템을 사용하는 동안 마운트된 상태로 유지된다는 것입니다. 파일 시스템의 모든 파일이 닫히고 모든 사용자와 프로세스가 마운트 지점 디렉터리에서 나가면 자동 마운터는 최소 시간 초과 후 파일 시스템을 마운트 해제합니다.

직접 및 간접 맵 사용 사례

자동 마운터는 직접 및 간접 마운트 지점 매핑을 모두 지원하여 두 가지 유형의 마운트 요청을 처리합니다. 직접 마운트는 파일 시스템이 변경되지 않는 알려진 마운트 지점 위치에 마운트되는 경우입니다. 자동 마운터에 대해 알아보기 전에 사용자가 구성한 거의 모든 파일 시스템 마운트는 직접 마운트의 예입니다. 직접 마운트 지점은 다른 일반 디렉터리와 마찬가지로 영구 디렉터리로 존재합니다.

간접 마운트는 마운트 요구가 발생할 때까지 마운트 지점 위치를 알 수 없는 경우입니다. 간접 마운트의 한 예는 원격 마운트 홈 디렉터리 구성으로, 사용자 홈 디렉터리의 디렉터리 경로에 사용자 이름이 포함됩니다. 자동 마운터에서 홈 디렉터리를 마운트하도록 지정한 사용자를 파악하고 사용할 마운트 지점 위치를 결정한 후에만 사용자의 원격 파일 시스템이 홈 디렉터리에 마운트됩니다. **autofs** 서비스는 간접 마운트 지점이 있는 것처럼 보여도 마운트 요청이 발생하면 해당 지점을 생성하고 요청이 종료되어 파일 시스템이 마운트 해제되면 다시 삭제합니다.

자동 마운터 서비스 구성

자동 마운트를 구성하는 프로세스에는 여러 단계가 있습니다.

먼저 **autofs** 및 **nfs-utils** 패키지를 설치해야 합니다.

```
[user@host ~]$ sudo dnf install autofs nfs-utils
```

이러한 패키지에는 NFS 내보내기에 자동 마운터를 사용하는 데 필요한 모든 요구 사항이 포함되어 있습니다.

마스터 맵 생성

그런 다음 마스터 맵 파일을 **/etc/auto.master.d**에 추가합니다. 이 파일은 마운트 지점의 기본 디렉터리를 확인하고 자동 마운트를 생성하는 매핑 파일을 확인합니다.

```
[user@host ~]$ sudo vim /etc/auto.master.d/demo.autofs
```

마스터 맵 파일의 이름은 대부분 임의적이며(일반적으로 의미가 있음), 하위 시스템의 경우 인식할 수 있도록 확장자가 **.autofs**여야 합니다. 단일 마스터 맵 파일에 여러 개의 항목을 배치할 수 있습니다. 또는 각 고유 항목이 논리적으로 그룹화된 마스터 맵 파일을 여러 개 만들 수 있습니다.

간접적으로 매핑된 마운트의 마스터 맵 항목에 다음 콘텐츠를 포함합니다.

```
/shares  /etc/auto.demo
```

이 항목은 간접 자동 마운트의 기반으로 **/shares** 디렉터리를 사용합니다. **/etc/auto.demo** 파일에는 마운트 세부 사항이 포함되어 있습니다. 절대 파일 이름을 사용하십시오. **autofs** 서비스를 시작하기 전에 **auto.demo** 파일을 생성해야 합니다.

간접 맵 생성

이제 매핑 파일을 생성합니다. 각 매핑 파일은 일련의 자동 마운트에 대한 마운트 지점, 마운트 옵션, 마운트 할 소스 위치를 확인합니다.

```
[user@host ~]$ sudo vim /etc/auto.demo
```

매핑 파일 이름 지정 규칙은 **/etc/auto.name**이며, 여기서 name에는 맵 콘텐츠가 반영됩니다.

```
work -rw, sync serverb:/shares/work
```

항목의 형식은 mount point, mount options, source location입니다. 이 예제는 간접 매핑 항목을 보여줍니다. 와일드카드를 사용하는 직접 맵과 간접 맵은 이 섹션의 뒷부분에서 다릅니다.

도움말 페이지에서 키로 알려진 **autofs** 서비스는 자동으로 마운트 지점을 생성하고 제거합니다. 이 경우 정규화된 마운트 지점은 **/shares/work**가 됩니다(마스터 맵 파일 참조). 필요한 경우 **autofs** 서비스에서 **/shares** 및 **/shares/work** 디렉터리를 생성 및 제거합니다.

이 예제에서 로컬 마운트 지점은 서버의 디렉터리 구조를 미러링합니다. 그러나 이러한 미러링은 필수가 아닙니다. 로컬 마운트 지점에는 임의의 이름을 사용할 수 있습니다. **autofs** 서비스는 클라이언트에 특정 이름 지정 구조를 적용하지 않습니다.

마운트 옵션은 대시 문자(-)로 시작하며 공백 없이 쉼표로 구분됩니다. 수동 마운트를 위한 파일 시스템 마운트 옵션은 자동 마운트 시에도 사용할 수 있습니다. 이 예제에서 자동 마운터는 읽기/쓰기 액세스 권한을 사용하여 내보내기를 마운트하고(**rw** 옵션), 쓰기 작업 중 서버가 즉시 동기화됩니다(**sync** 옵션).

유능한 automounter-specific 옵션 포함 **-fstype=**과 **-strict .fstype**은 파일 시스템 유형을 **nfs4** 또는 **xfs** 등으로 지정하고 **strict**는 파일 시스템을 마운트할 때 치명적인 오류를 해결하는 데 사용합니다.

NFS 내보내기의 소스 위치는 **host:/ pathname** 패턴을 따르며, 이 예제의 경우 **serverb:/shares/work**입니다. 이 자동 마운트가 성공하려면 NFS 서버 **serverb**에서 읽기/쓰기 액세스 권한을 사용하여 디렉터리를 내보내야 하고, 액세스 권한을 요청하는 사용자에게 디렉터리에 대한 표준 Linux 파일 권한이 있어야 합니다. **serverb**가 읽기 전용 액세스 권한이 있는 디렉터리를 내보내는 경우, 클라이언트는 읽기/쓰기 액세스 권한을 요청하더라도 읽기 전용 액세스 권한만 갖게 됩니다.

간접 맵의 와일드카드

NFS 서버가 디렉터리 내에서 여러 하위 디렉터리를 내보내는 경우 자동 마운터가 단일 매핑 항목을 사용하여 그중 임의의 하위 디렉터리에 액세스하도록 구성할 수 있습니다.

앞의 예제에서 **serverb:/shares**를 두 개 이상의 하위 디렉터리로 내보내고 동일한 마운트 옵션을 사용하여 액세스 가능한 경우 **/etc/auto.demo** 파일의 콘텐츠는 다음과 같을 것입니다.

```
* -rw, sync serverb:/shares/&
```

마운트 지점(또는 키)은 별표 문자(*)이며 소스 위치의 하위 디렉터리는 앤퍼샌드 문자(&)입니다. 항목의 나머지는 동일합니다.

사용자가 **/shares/work**에 액세스를 시도하면 소스 위치의 앤퍼샌드가 * 키(이 예제에서 **work**)로 바뀌고, **serverb:/exports/work**가 마운트됩니다. 간접 예제와 마찬가지로 **autofs** 서비스는 **work** 디렉터리를 자동으로 생성 및 제거합니다.

직접 맵 생성

직접 맵은 NFS 내보내기를 절대 경로 마운트 지점에 매핑하는 데 사용됩니다. 직접 맵 파일은 하나만 필요하며 여러 개의 직접 맵을 포함할 수 있습니다.

직접 매핑된 마운트 지점을 사용할 수 있도록 마스터 맵 파일이 다음과 같이 표시될 수 있습니다.

```
/- /etc/auto.direct
```

모든 직접 맵 항목은 기본 디렉터리로 **/-**를 사용합니다. 이 경우 마운트 상세 정보가 포함된 매핑 파일은 **/etc/auto.direct**입니다.

`/etc/auto.direct` 파일의 내용은 다음과 같을 수 있습니다.

```
/mnt/docs -rw, sync server:/shares/docs
```

마운트 지점(또는 키)은 항상 절대 경로입니다. 나머지 매핑 파일에서는 동일한 구조를 사용합니다.

이 예에서는 `/mnt` 디렉터리가 있고 이 디렉터리는 `autofs` 서비스에서 관리하지 않습니다. `autofs` 서비스는 전체 `/mnt/docs` 디렉터리를 자동으로 생성 및 제거합니다.

자동 마운터 서비스 시작

마지막으로 `systemctl` 명령을 사용하여 `autofs` 서비스를 시작하고 활성화합니다.

```
[user@host ~]$ sudo systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/
lib/systemd/system/autofs.service.
```

대체 `systemd.automount` 메서드

`systemd` 데몬은 `x-systemd.automount` 옵션을 포함하는 `/etc/fstab` 파일의 항목에 대한 유닛 파일을 자동으로 생성할 수 있습니다. 항목의 마운트 옵션을 수정한 후 `systemctl daemon-reload` 명령을 사용하여 새 유닛 파일을 생성한 다음 `systemctl start unit.automount` 명령을 사용하여 해당 자동 마운트 구성을 활성화합니다.

유닛의 이름은 마운트 위치에 따라 지정됩니다. 예를 들어 마운트 지점이 `/remote/finance`이면 유닛 파일의 이름은 `remote-finance.automount`입니다. `systemd` 데몬은 `/remote/finance` 디렉터리에 처음 액세스할 때 파일 시스템을 마운트합니다.

이 방법은 `autofs` 서비스를 설치하고 구성하는 것보다 더 간단할 수 있습니다. 그러나 `systemd.automount` 유닛은 `autofs` 직접 맵과 유사하게 절대 경로 마운트 지점만 지원할 수 있습니다.



참조

`autofs(5)`, `automount(8)`, `auto.master(5)`, `mount.nfs(8)`,
`systemd.automount(5)` 도움말 페이지

▶ 연습 가이드

네트워크 연결 스토리지 자동 마운트

이 연습에서는 NFS 파일 시스템을 마운트하는 직접 매핑 및 간접 매핑된 자동 관리 마운트 지점을 생성합니다.

결과

- 자동 마운터의 필수 패키지를 설치합니다.
- 사전 구성된 NFSv4 서버에 있는 리소스를 사용하여 직접 및 간접 자동 마운터 맵을 구성합니다.
- 직접 및 간접 자동 마운터 맵의 차이점을 설명합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 시작 스크립트는 네트워크에서 **servera** 및 **serverb**에 연결할 수 있는지 확인합니다. 그러한 서버를 사용할 수 없는 경우 스크립트에서 알려줍니다. 시작 스크립트가 **serverb**를 NFSv4 서버로 구성하고, 권한을 설정한 후 디렉터리를 내보냅니다. 이 스크립트는 또한 **servera** 및 **serverb**에서 필요한 사용자와 그룹을 생성합니다.

```
[student@workstation ~]$ lab start netstorage-autofs
```

지침

인터넷 서비스 프로바이더는 온디맨드로 제공해야 하는 중요 문서가 포함된 공유 디렉터리를 호스팅하는데 중앙 서버 **serverb**를 사용합니다. 사용자는 **servera**에 로그인할 때 자동 마운트된 공유 디렉터리에 대한 액세스 권한이 있어야 합니다.

다음 목록은 이 연습을 완료하기 위한 환경 특성을 제공합니다.

- serverb** 시스템은 **west**, **central**, **east** 하위 디렉터리가 차례로 포함된 **/shares/indirect** 디렉터리를 내보냅니다.
- serverb** 시스템은 **/shares/direct/external** 디렉터리도 내보냅니다.
- operators** 그룹은 **operator1** 및 **operator2** 사용자로 구성됩니다. 이 그룹은 내보낸 **/shares/indirect/west**, **/shares/indirect/central**, **/shares/indirect/east** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- contractors** 그룹은 **contractor1** 및 **contractor2** 사용자로 구성됩니다. 해당 사용자는 내보낸 **/shares/direct/external** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- servera**의 예상 마운트 지점은 **/external** 및 **/internal**입니다.
- 내보낸 **/shares/direct/external** 디렉터리는 **/external**의 직접 맵을 사용하여 **servera**에 자동으로 마운트됩니다.
- 내보낸 **/shares/indirect/west** 디렉터리는 **/internal/west**의 간접 맵을 사용하여 **servera**에 자동으로 마운트됩니다.

- 내보낸 `/shares/indirect/central` 디렉터리는 `/internal/central`의 간접 맵을 사용하여 `servera`에 자동으로 마운트됩니다.
- 내보낸 `/shares/indirect/east` 디렉터리는 `/internal/east`의 간접 맵을 사용하여 `servera`에 자동으로 마운트됩니다.
- 모든 사용자 암호는 `redhat`으로 설정되어 있습니다.
- `nfs-utils` 패키지가 이미 설치되어 있습니다.

▶ 1. `servera`에 로그인하여 필요한 패키지를 설치합니다.

1.1. `servera`에 `student` 사용자로 로그인한 후 `root` 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

1.2. `autofs` 패키지를 설치합니다.

```
[root@servera ~]# dnf install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

▶ 2. `serverb`에서 내보내기를 사용하여 `servera`에 자동 마운터 직접 맵을 구성합니다. 마스터 맵의 경우 파일 `/etc/auto.master.d/direct.autofs`, 매핑 파일의 경우 `/etc/auto.direct`라는 파일을 사용하여 직접 맵을 만듭니다. `/external` 디렉터리를 `servera`의 주 마운트 지점으로 사용합니다.

2.1. 자동 마운터를 구성하기 전에 NFS 서버 및 내보내기를 테스트합니다.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/direct/external /mnt
[root@servera ~]# ls -l /mnt
total 4
-rw-r--r-- 1 root contractors 22 Apr 7 23:15 README.txt
[root@servera ~]# umount /mnt
```

2.2. `/etc/auto.master.d/direct.autofs`라는 마스터 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
/- /etc/auto.direct
```

2.3. `/etc/auto.direct`라는 직접 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
/external -rw, sync, fstype=nfs4 serverb.lab.example.com:/shares/direct/external
```

- ▶ 3. **serverb**에서 내보내기를 사용하여 **servera**에 자동 마운터 간접 맵을 구성합니다. 마스터 맵의 경우 파일 **/etc/auto.master.d/indirect.autofs**, 매핑 파일의 경우 **/etc/auto.indirect**라는 파일을 사용하여 간접 맵을 만듭니다. **/internal** 디렉터리를 **servera**의 주 마운트 지점으로 사용합니다.

3.1. 자동 마운터를 구성하기 전에 NFS 서버 및 내보내기를 테스트합니다.

```
[root@servera ~]# mount -t nfs \
serverb.lab.example.com:/shares/indirect /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrws---. 2 root operators 24 Apr  7 23:34 central
drwxrws---. 2 root operators 24 Apr  7 23:34 east
drwxrws---. 2 root operators 24 Apr  7 23:34 west
[root@servera ~]# umount /mnt
```

3.2. **/etc/auto.master.d/indirect.autofs**라는 마스터 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
/internal /etc/auto.indirect
```

3.3. **/etc/auto.indirect**라는 간접 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
* -rw, sync, fstype=nfs4 serverb.lab.example.com:/shares/indirect/&
```

- ▶ 4. **servera**에서 **autofs** 서비스를 시작하고 부팅 시 자동으로 시작되도록 활성화합니다.

4.1. **autofs**에서 **servera** 서비스를 시작하고 활성화합니다.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/
lib/systemd/system/autofs.service.
```

- ▶ 5. **contractor1** 사용자로 직접 자동 마운터 맵을 테스트합니다. 완료되면 **contractor1**에서 **servera** 사용자 세션을 종료합니다.

5.1. **contractor1** 사용자로 전환합니다.

```
[root@servera ~]# su - contractor1
[contractor1@servera ~]$
```

5.2. **/external** 마운트 지점을 나열합니다.

```
[contractor1@servera ~]$ ls -l /external
total 4
-rw-r--r--. 1 root contractors 22 Apr  7 23:34 README.txt
```

5.3. 콘텐츠를 검토하고 **/external** 마운트 지점에 대한 액세스를 테스트합니다.

```
[contractor1@servera ~]$ cat /external/README.txt
###External Folder##
[contractor1@servera ~]$ echo testing-direct > /external/testing.txt
[contractor1@servera ~]$ cat /external/testing.txt
testing-direct
```

5.4. **contractor1** 사용자 세션을 종료합니다.

```
[contractor1@servera ~]$ exit
logout
[root@servera ~]#
```

▶ 6. **operator1** 사용자로 간접 자동 마운터 맵을 테스트합니다. 완료되면 **servera**에서 로그아웃합니다.

6.1. **operator1** 사용자로 전환합니다.

```
[root@servera ~]# su - operator1
[operator1@servera ~]$
```

6.2. **/internal** 마운트 지점을 나열합니다.

```
[operator1@servera ~]$ ls -l /internal
total 0
```



참고

자동 마운터 간접 맵을 사용하면 내보낸 각 하위 디렉터리에 액세스하여 마운트해야 합니다.
자동 마운터 직접 맵을 사용하면 매핑된 마운트 지점에 액세스한 직후부터 내보낸 디렉터리의 하위 디렉터리 및 콘텐츠를 보고 액세스할 수 있습니다.

6.3. 자동 마운터로 내보낸 **/internal/west** 디렉터리 액세스를 테스트합니다.

```
[operator1@servera ~]$ ls -l /internal/west/
total 4
-rw-r--r-- 1 root operators 18 Apr  7 23:34 README.txt
[operator1@servera ~]$ cat /internal/west/README.txt
###West Folder###
[operator1@servera ~]$ echo testing-1 > /internal/west/testing-1.txt
[operator1@servera ~]$ cat /internal/west/testing-1.txt
testing-1
[operator1@servera ~]$ ls -l /internal
total 0
drwxrws--- 2 root operators 24 Apr  7 23:34 west
```

6.4. 자동 마운터로 내보낸 **/internal/central** 디렉터리 액세스를 테스트합니다.

```
[operator1@servera ~]$ ls -l /internal/central
total 4
-rw-r--r-- 1 root operators 21 Apr  7 23:34 README.txt
```

```
[operator1@servera ~]$ cat /internal/central/README.txt  
###Central Folder###  
[operator1@servera ~]$ echo testing-2 > /internal/central/testing-2.txt  
[operator1@servera ~]$ cat /internal/central/testing-2.txt  
testing-2  
[operator1@servera ~]$ ls -l /internal  
total 0  
drwxrws--- 2 root operators 24 Apr  7 23:34 central  
drwxrws--- 2 root operators 24 Apr  7 23:34 west
```

6.5. 자동 마운터로 내보낸 `/internal/east` 디렉터리 액세스를 테스트합니다.

```
[operator1@servera ~]$ ls -l /internal/east  
total 4  
-rw-r--r-- 1 root operators 18 Apr  7 23:34 README.txt  
[operator1@servera ~]$ cat /internal/east/README.txt  
###East Folder###  
[operator1@servera ~]$ echo testing-3 > /internal/east/testing-3.txt  
[operator1@servera ~]$ cat /internal/east/testing-3.txt  
testing-3  
[operator1@servera ~]$ ls -l /internal  
total 0  
drwxrws--- 2 root operators 24 Apr  7 23:34 central  
drwxrws--- 2 root operators 24 Apr  7 23:34 east  
drwxrws--- 2 root operators 24 Apr  7 23:34 west
```

6.6. 자동 마운터로 내보낸 `/external` 디렉터리 액세스를 테스트합니다.

```
[operator1@servera ~]$ ls -l /external  
ls: cannot open directory '/external': Permission denied
```

6.7. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[operator1@servera ~]$ exit  
logout  
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netstorage-autofs
```

이것으로 섹션을 완료합니다.

▶ 랩

네트워크 연결 스토리지 액세스

이 랩에서는 NFSv4 서버의 내보내기를 사용하여 간접 맵으로 자동 마운터를 구성합니다.

결과

- 자동 마운터를 설정하는 데 필요한 필수 패키지를 설치합니다.
- 사전 구성된 NFSv4 서버에 있는 리소스를 사용하여 자동 마운터 간접 맵을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 시작 스크립트는 네트워크에서 **servera** 및 **serverb** 시스템에 연결할 수 있는지 확인합니다. 시작 스크립트가 **serverb**를 NFSv4 서버로 구성하고, 권한을 설정한 후 디렉터리를 내보냅니다. 이 스크립트는 또한 **servera** 및 **serverb** 시스템에서 필요한 사용자와 그룹을 생성합니다.

```
[student@workstation ~]$ lab start netstorage-review
```

지침

한 IT 지원 회사에서 자사의 그룹 및 사용자를 위해 중앙 서버(**serverb**)를 사용하여 일부 내보낸 디렉터리를 **/shares**에서 호스팅합니다. 사용자는 로그인한 다음 내보낸 디렉터리를 온디맨드로 마운트하여 **servera**의 **/remote** 디렉터리에서 사용할 수 있도록 준비해야 합니다.

다음 목록은 이 연습을 완료하기 위한 환경 특성을 제공합니다.

- serverb** 시스템은 **management**, **production**, **operation** 하위 디렉터리가 차례로 포함된 **/shares** 디렉터리를 공유합니다.
- managers** 그룹은 **manager1** 및 **manager2** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/management** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- production** 그룹은 **dbuser1** 및 **sysadmin1** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/production** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- operators** 그룹은 **contractor1** 및 **consultant1** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/operation** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- servera**의 주 마운트 지점은 **/remote** 디렉터리입니다.
- /etc/auto.master.d/shares.autofs** 파일을 마스터 맵 파일로 사용하고 **/etc/auto.shares** 파일을 간접 맵 파일로 사용합니다.
- 내보낸 **/shares/management** 디렉터리는 **servera**의 **/remote/management**에 자동으로 마운트됩니다.
- 내보낸 **/shares/production** 디렉터리는 **servera**의 **/remote/production**에 자동으로 마운트됩니다.

- 내보낸 `/shares/operation` 디렉터리는 `servera`의 `/remote/operation`에 자동으로 마운트 됩니다.
 - 모든 사용자 암호는 `redhat`으로 설정되어 있습니다.
- `servera`에 로그인하여 필요한 패키지를 설치합니다.
 - `serverb`에서 내보내기를 사용하여 `servera`에 자동 마운터 간접 맵을 구성합니다. 마스터 맵의 경우 파일 `/etc/auto.master.d/shares.autofs`, 매핑 파일의 경우 `/etc/auto.shares`라는 파일을 사용하여 간접 맵을 만듭니다. `/remote` 디렉터리를 `servera`의 주 마운트 지점으로 사용 합니다. `servera`를 재부팅하여 `autofs` 서비스가 자동으로 시작되는지 확인합니다.
 - 다양한 사용자가 포함된 `autofs` 구성을 테스트합니다. 완료되면 `servera`에서 로그아웃합니다.

평가

`workstation` 시스템에서 `lab` 명령을 실행하여 이 연습의 성공 상태를 확인합니다.

```
[student@workstation ~]$ lab grade netstorage-review
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netstorage-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

네트워크 연결 스토리지 액세스

이 랩에서는 NFSv4 서버의 내보내기를 사용하여 간접 맵으로 자동 마운터를 구성합니다.

결과

- 자동 마운터를 설정하는 데 필요한 필수 패키지를 설치합니다.
- 사전 구성된 NFSv4 서버에 있는 리소스를 사용하여 자동 마운터 간접 맵을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 시작 스크립트는 네트워크에서 **servera** 및 **serverb** 시스템에 연결할 수 있는지 확인합니다. 시작 스크립트가 **serverb**를 NFSv4 서버로 구성하고, 권한을 설정한 후 디렉터리를 내보냅니다. 이 스크립트는 또한 **servera** 및 **serverb** 시스템에서 필요한 사용자와 그룹을 생성합니다.

```
[student@workstation ~]$ lab start netstorage-review
```

지침

한 IT 지원 회사에서 자사의 그룹 및 사용자를 위해 중앙 서버(**serverb**)를 사용하여 일부 내보낸 디렉터리를 **/shares**에서 호스팅합니다. 사용자는 로그인한 다음 내보낸 디렉터리를 온디맨드로 마운트하여 **servera**의 **/remote** 디렉터리에서 사용할 수 있도록 준비해야 합니다.

다음 목록은 이 연습을 완료하기 위한 환경 특성을 제공합니다.

- serverb** 시스템은 **management**, **production**, **operation** 하위 디렉터리가 차례로 포함된 **/shares** 디렉터리를 공유합니다.
- managers** 그룹은 **manager1** 및 **manager2** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/management** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- production** 그룹은 **dbuser1** 및 **sysadmin1** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/production** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- operators** 그룹은 **contractor1** 및 **consultant1** 사용자로 구성됩니다. 그러한 사용자는 내보낸 **/shares/operation** 디렉터리에 대한 읽기 및 쓰기 액세스 권한이 있습니다.
- servera**의 주 마운트 지점은 **/remote** 디렉터리입니다.
- /etc/auto.master.d/shares.autofs** 파일을 마스터 맵 파일로 사용하고 **/etc/auto.shares** 파일을 간접 맵 파일로 사용합니다.
- 내보낸 **/shares/management** 디렉터리는 **servera**의 **/remote/management**에 자동으로 마운트됩니다.
- 내보낸 **/shares/production** 디렉터리는 **servera**의 **/remote/production**에 자동으로 마운트됩니다.

- 내보낸 `/shares/operation` 디렉터리는 `servera`의 `/remote/operation`에 자동으로 마운트 됩니다.
- 모든 사용자 암호는 `redhat`으로 설정되어 있습니다.

1. `servera`에 로그인하여 필요한 패키지를 설치합니다.

- `servera`에 `student` 사용자로 로그인한 후 `root` 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- `autoofs` 패키지를 설치합니다.

```
[root@servera ~]# dnf install autoofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. `serverb`에서 내보내기를 사용하여 `servera`에 자동 마운터 간접 맵을 구성합니다. 마스터 맵의 경우 파일 `/etc/auto.master.d/shares.autoofs`, 매핑 파일의 경우 `/etc/auto.shares`라는 파일을 사용하여 간접 맵을 만듭니다. `/remote` 디렉터리를 `servera`의 주 마운트 지점으로 사용합니다. `servera`를 재부팅하여 `autoofs` 서비스가 자동으로 시작되는지 확인합니다.

- 자동 마운터를 구성하기 전에 NFS 서버를 테스트합니다.

```
[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares /mnt
[root@servera ~]# ls -l /mnt
total 0
drwxrwx---. 2 root managers 25 Apr 4 01:13 management
drwxrwx---. 2 root operators 25 Apr 4 01:13 operation
drwxrwx---. 2 root production 25 Apr 4 01:13 production
[root@servera ~]# umount /mnt
```

- `/etc/auto.master.d/shares.autoofs`라는 마스터 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
/remote /etc/auto.shares
```

- `/etc/auto.shares`라는 간접 맵 파일을 만들고 다음 내용을 삽입한 후 변경 사항을 저장합니다.

```
* -rw, sync, fstype=nfs4 serverb.lab.example.com:/shares/&
```

- `autoofs`에서 `servera` 서비스를 시작하고 활성화합니다.

```
[root@servera ~]# systemctl enable --now autofs
Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/
lib/systemd/system/autofs.service.
```

3. 다양한 사용자가 포함된 **autofs** 구성 테스트합니다. 완료되면 **servera**에서 로그아웃합니다.

- 3.1. **manager1** 사용자로 전환하고 액세스를 테스트합니다.

```
[root@servera ~]# su - manager1
[manager1@servera ~]$ ls -l /remote/management/
total 4
-rw-r--r--. 1 root managers 46 Apr  4 01:13 Welcome.txt
[manager1@servera ~]$ cat /remote/management>Welcome.txt
###Welcome to Management Folder on SERVERB###
[manager1@servera ~]$ echo TEST1 > /remote/management/Test.txt
[manager1@servera ~]$ cat /remote/management/Test.txt
TEST1
[manager1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[manager1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[manager1@servera ~]$ exit
logout
[root@servera ~]#
```

- 3.2. **dbuser1** 사용자로 전환하고 액세스를 테스트합니다.

```
[root@servera ~]# su - dbuser1
[dbuser1@servera ~]$ ls -l /remote/production/
total 4
-rw-r--r--. 1 root production 46 Apr  4 01:13 Welcome.txt
[dbuser1@servera ~]$ cat /remote/production>Welcome.txt
###Welcome to Production Folder on SERVERB###
[dbuser1@servera ~]$ echo TEST2 > /remote/production/Test.txt
[dbuser1@servera ~]$ cat /remote/production/Test.txt
TEST2
[dbuser1@servera ~]$ ls -l /remote/operation/
ls: cannot open directory '/remote/operation/': Permission denied
[dbuser1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[dbuser1@servera ~]$ exit
logout
[root@servera ~]#
```

- 3.3. **contractor1** 사용자로 전환하고 액세스를 테스트합니다.

```
[root@servera ~]# su - contractor1
[contractor1@servera ~]$ ls -l /remote/operation/
total 4
-rw-r--r--. 1 root operators 45 Apr  4 01:13 Welcome.txt
[contractor1@servera ~]$ cat /remote/operation>Welcome.txt
###Welcome to Operation Folder on SERVERB###
```

14장 | 네트워크 연결 스토리지 액세스

```
[contractor1@servera ~]$ echo TEST3 > /remote/operation/Test.txt
[contractor1@servera ~]$ cat /remote/operation/Test.txt
TEST3
[contractor1@servera ~]$ ls -l /remote/management/
ls: cannot open directory '/remote/management/': Permission denied
[contractor1@servera ~]$ ls -l /remote/production/
ls: cannot open directory '/remote/production/': Permission denied
[contractor1@servera ~]$ exit
logout
[root@servera ~]#
```

3.4. NFS 자동 마운트 내보내기에 대한 **mount** 옵션을 살펴봅니다.

```
[root@servera ~]# mount | grep nfs
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
serverb.lab.example.com:/shares/management on /remote/management type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/operation on /remote/operation type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
serverb.lab.example.com:/shares/production on /remote/production type nfs4
(rw,relatime,vers=4.2,rsize=262144,wsize=262144,namlen=255,
sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
local_lock=none,addr=172.25.250.11)
```

3.5. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
```

평가

workstation 시스템에서 **lab** 명령을 실행하여 이 연습의 성공 상태를 확인합니다.

```
[student@workstation ~]$ lab grade netstorage-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netstorage-review
```

이것으로 섹션을 완료합니다.

요약

- 명령줄에서 NFS 공유를 마운트 및 마운트 해제합니다.
- 시작 시 자동으로 마운트하도록 NFS 공유를 구성합니다.
- 직접 및 간접 맵을 사용하는 자동 마운터를 구성하고 그 차이를 설명합니다.

네트워크 보안 관리

목적

시스템 방화벽 및 SELinux 규칙을 사용하여 서비스에 대한 네트워크 연결을 제어합니다.

목표

- `firewalld` 규칙을 사용하여 시스템 서비스에 대한 네트워크 연결을 수락하거나 거부합니다.
- 네트워크 포트에 바인딩할 서비스에 맞는 올바른 SELinux 유형이 있는지 확인합니다.

섹션

- 서버 방화벽 관리(안내에 따른 연습)
- SELinux 포트 레이블 지정 제어(안내에 따른 연습)

랩

네트워크 보안 관리

서버 방화벽 관리

목표

firewalld 규칙을 사용하여 시스템 서비스에 대한 네트워크 연결을 수락하거나 거부합니다.

방화벽 아키텍처의 개념

Linux 커널은 패킷 필터링, 네트워크 주소 변환, 포트 변환과 같은 네트워크 트래픽 작업에 필요한 **netfilter** 프레임워크를 제공합니다. **netfilter** 프레임워크에는 커널 모듈이 시스템의 네트워크 스택을 통과하는 네트워크 패킷과 상호 작용하기 위한 후크가 포함되어 있습니다. 기본적으로 **netfilter** 후크는 이벤트(예: 인터페이스에 들어오는 패킷)를 가로채고 다른 관련 루틴(예: 방화벽 규칙)을 실행하는 커널 루틴입니다.

nftables 프레임워크

nftables 패킷 분류 프레임워크는 **netfilter** 프레임워크를 기반으로 빌드되어 네트워크 트래픽에 방화벽 규칙을 적용합니다. Red Hat Enterprise Linux 9에서 **nftables** 프레임워크는 시스템 방화벽의 코어로, 더 이상 사용되지 않는 **iptables** 프레임워크를 대체합니다.

nftables 프레임워크는 **iptables**에 비해 향상된 유용성과 더 효율적인 규칙 집합 등 다양한 이점을 제공합니다. 예를 들어 **iptables** 프레임워크에는 프로토콜별로 규칙이 필요하지만, **nftables** 규칙은 IPv4 및 IPv6 트래픽에 동시에 적용할 수 있습니다. **iptables** 프레임워크에는 각 프로토콜에 대해 **iptables**, **ip6tables**, **arptables** 및 **ebtables**와 같은 다양한 도구를 사용해야 합니다. 반면 **nftables** 프레임워크는 단일 **nft** 사용자 공간 유ти리티를 사용하여 단일 인터페이스를 통해 모든 프로토콜을 관리합니다.



참고

iptables-translate 및 **ip6tables-translate** 유ти리티를 사용하여 이전 **iptables** 구성 파일을 해당하는 **nftables** 구성 파일로 변환합니다.

firewalld 서비스

firewalld 서비스는 동적 방화벽 관리자이며 **nftables** 프레임워크에 권장되는 프런트엔드입니다. Red Hat Enterprise Linux 9 배포에는 **firewalld** 패키지가 포함되어 있습니다.

firewalld 서비스는 네트워크 트래픽을 영역으로 분류하여 방화벽 관리를 간소화합니다. 할당된 네트워크 패킷 영역은 패킷의 소스 IP 주소 또는 들어오는 네트워크 인터페이스와 같은 기준에 따라 다릅니다. 각 영역에는 열려 있거나 닫혀 있는 포트 및 서비스의 자체 목록이 있습니다.



참고

네트워크를 자주 변경하는 노트북 또는 기타 시스템의 경우 **NetworkManager** 서비스에서 연결의 방화벽 영역을 자동으로 설정할 수 있습니다. 이 서비스는 집, 직장, 공용 무선 네트워크를 스위칭할 때 유용합니다. 사용자는 집과 회사 네트워크에서는 시스템의 **sshd** 서비스에 연결하지만 근처 커피숍의 공용 무선 네트워크에서는 연결하고 싶지 않을 수 있습니다.

firewalld 서비스는 시스템에 들어오는 모든 패킷의 소스 주소를 확인합니다. 해당 소스 주소가 특정 영역에 할당되면 해당 영역의 규칙이 적용됩니다. 소스 주소가 영역에 할당되지 않으면 **firewalld** 서비스가 패킷을 들어오는 네트워크 인터페이스의 영역과 연결하며 해당 영역의 규칙이 적용됩니다. 네트워크 인터페이스가 영역과 연결되지 않은 경우에는 **firewalld** 서비스가 패킷을 기본 영역으로 전송합니다.

기본 영역은 별도의 영역이 아닌 기존 영역에 할당된 대상을 가리킵니다. **firewalld** 서비스는 처음에 **public** 영역을 기본으로 지정하고 **lo** 루프백 인터페이스를 **trusted** 영역에 매핑합니다.

대부분의 영역에서는 트래픽이 특정 포트와 프로토콜 목록(예: **631/udp**) 또는 사전 정의된 서비스 구성(예: **ssh**)과 일치하는 방화벽을 통과합니다. 일반적으로 허용된 포트 및 프로토콜 또는 서비스와 일치하지 않는 트래픽은 거부됩니다. (기본적으로 모든 트래픽을 허용하는 **trusted** 영역은 예외입니다.)

사전 정의된 영역

firewalld 서비스는 사용자 지정할 수 있는 사전 정의된 영역을 사용합니다. 기본적으로 모든 영역은 시스템에서 시작한 기존 세션에 포함된 들어오는 트래픽을 허용하며 나가는 트래픽도 모두 허용합니다. 다음 표에 초기 영역 구성이 자세히 설명되어 있습니다.

firewalld 영역의 기본 구성

영역 이름	기본 구성
trusted	들어오는 모든 트래픽을 허용합니다.
home	나가는 트래픽과 관련이 없거나 ssh , mdns , ipp-client , samba-client 또는 dhcpv6-client 사전 정의 서비스와 일치하지 않을 경우 들어오는 트래픽을 거부합니다.
internal	나가는 트래픽과 관련이 없거나 ssh , mdns , ipp-client , samba-client 또는 dhcpv6-client 사전 정의 서비스(시작하는 home 영역과 동일)와 일치하지 않을 경우 들어오는 트래픽을 거부합니다.
work	나가는 트래픽과 관련이 없거나 ssh , ipp-client 또는 dhcpv6-client 사전 정의 서비스와 일치하지 않을 경우 들어오는 트래픽을 거부합니다.
public	나가는 트래픽과 관련이 없거나 ssh 또는 dhcpv6-client 사전 정의 서비스와 일치하지 않을 경우 들어오는 트래픽을 거부합니다. 새로 추가된 네트워크 인터페이스의 기본 영역입니다.
external	나가는 트래픽과 일치하지 않거나 ssh 사전 정의 서비스와 일치하지 않을 경우 들어오는 트래픽을 거부합니다. 이 영역을 통해 전달되어 나가는 IPv4 트래픽은 나가는 네트워크 인터페이스의 IPv4 주소에서 시작된 것처럼 마스커레이드 됩니다.
dmz	나가는 트래픽과 일치하지 않거나 ssh 사전 정의 서비스와 일치하지 않을 경우 들어오는 트래픽을 거부합니다.
block	나가는 트래픽과 관련되지 않은 경우 들어오는 모든 트래픽을 거부합니다.
drop	나가는 트래픽과 관련되지 않은 경우(ICMP 오류에도 대응하지 않음) 들어오는 모든 트래픽을 드롭합니다.

사용 가능한 사전 정의 영역과 의도된 용도 목록은 **firewalld.zones**(5) 도움말 페이지에서 확인하십시오.

사전 정의 서비스

firewalld 서비스에는 방화벽 규칙 설정을 간소화하기 위해 일반 서비스에 대한 사전 정의된 구성이 포함되어 있습니다. 예를 들어 NFS 서버의 관련 포트를 조사하는 대신 사전 정의된 **nfs** 구성은 사용하여 올바른 포트 및 프로토콜에 대한 규칙을 생성합니다. 다음 표에는 기본 **firewalld** 영역에서 활성화되어 있을 수 있는 몇 가지 사전 정의된 서비스 구성이 나열되어 있습니다.

선택된 사전 정의 Firewalld 서비스

서비스 이름	구성
ssh	로컬 SSH 서버. 22/tcp로 가는 트래픽.
dhcpv6-client	로컬 DHCPv6 클라이언트. fe80::/64 IPv6 네트워크의 546/udp로 가는 트래픽.
ipp-client	로컬 IPP 인쇄. 631/udp에 대한 트래픽.
samba-client	로컬 Windows 파일 및 인쇄 공유 클라이언트. 137/udp 및 138/udp에 대한 트래픽.
mdns	mDNS(멀티캐스트 DNS) 로컬 링크 이름 확인. 224.0.0.251(IPv4) 또는 ff02::fb(IPv6) 멀티캐스트 주소로 보내는 5353/udp에 대한 트래픽.
cockpit	로컬 및 원격 시스템을 관리하고 모니터링하는 Red Hat Enterprise Linux 웹 기반 인터페이스입니다. 9090 포트로의 트래픽입니다.

firewalld 패키지에는 사전 정의된 많은 서비스 구성이 포함되어 있습니다. **firewall-cmd --get-services** 명령을 사용하여 서비스를 나열할 수 있습니다.

```
[root@host ~]# firewall-cmd --get-services
RH-Satellite-6 RH-Satellite-6-capsule amanda-client amanda-k5-client amqp amqps
apcupsd audit bacula bacula-client bb bgp bitcoin bitcoin-rpc bitcoin-testnet
bitcoin-testnet-rpc bittorrent-lsd ceph ceph-mon cfengine cockpit collectd
...output omitted...
```

사전 정의된 서비스 구성이 시나리오에 적합하지 않은 경우 필요한 포트 및 프로토콜을 수동으로 지정할 수 있습니다. 웹 콘솔 그래픽 인터페이스를 사용하여 사전 정의된 서비스를 검토하고 추가 포트 및 프로토콜을 수동으로 정의할 수 있습니다.

firewalld 데몬 구성

특히 시스템 관리자가 **firewalld** 서비스와 상호 작용하는 데 사용하는 두 가지 일반적인 방법은 다음과 같습니다.

- 웹 콘솔 그래픽 인터페이스
- **firewall-cmd** 명령줄 도구

웹 콘솔을 사용하여 방화벽 서비스 구성

웹 콘솔을 사용하여 방화벽 서비스를 관리하려면 로그인하여 권한을 에스컬레이션해야 합니다. **Limited access** 또는 **Turn on administrative access** 버튼을 클릭하여 권한을 에스컬레이션할 수 있습니다. 그런 다음 메시지가 표시되면 암호를 입력합니다. 관리 모드는 사용자의 sudo 구성에 따라 권한을 에스컬레이션 합니다. 보안을 위해 시스템에서 관리 권한이 필요한 작업을 수행한 후에는 제한된 액세스 모드로 다시 전환해야 합니다.

왼쪽 탐색 메뉴에서 **Networking** 옵션을 클릭하여 기본 네트워킹 페이지에 **Firewall** 섹션을 표시합니다. **Edit rules and zones** 버튼 영역을 클릭하여 **Firewall** 페이지로 이동합니다.

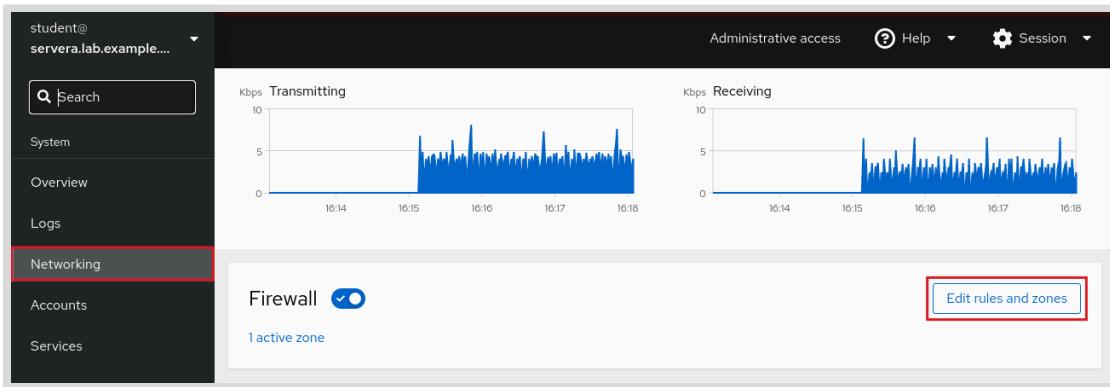


그림 15.1: 웹 콘솔 네트워킹 페이지

Firewall 페이지에는 활성 영역과 허용된 서비스가 표시됩니다. 서비스 세부 정보를 보려면 서비스 이름 왼쪽에 있는 화살표(>)를 클릭합니다. 영역에 서비스를 추가하려면 해당 영역의 오른쪽 상단에 있는 **Add services** 버튼을 클릭합니다.

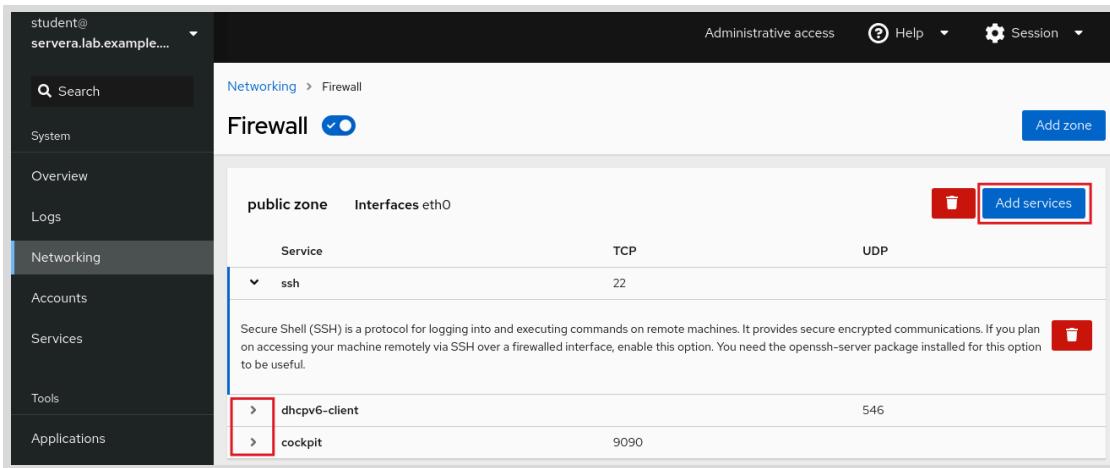


그림 15.2: 웹 콘솔 방화벽 페이지

Add Services 페이지에 사용 가능한 사전 정의 서비스가 표시됩니다.

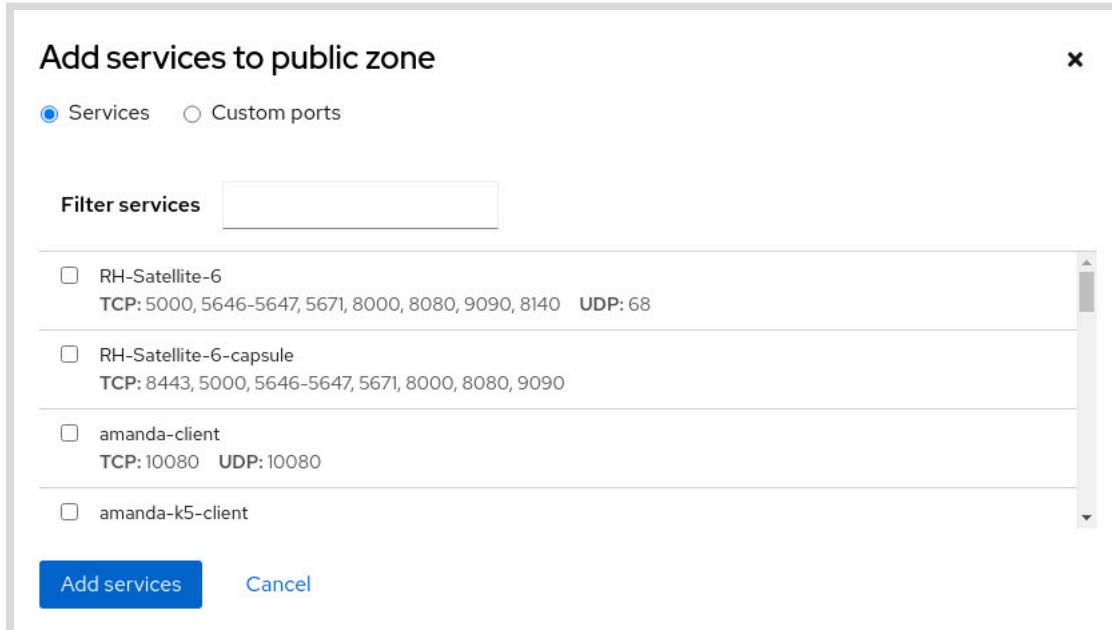


그림 15.3: 웹 콘솔에서 서비스 메뉴 추가

서비스를 선택하려면 목록을 스크롤하거나 **Filter services** 텍스트 상자에 선택 사항을 입력합니다. 다음 예제에서 **http** 문자열은 웹 관련 서비스로 필터링합니다. 방화벽을 통과하도록 하용하려면 서비스 왼쪽에 있는 확인란을 선택합니다. **Add services** 버튼을 클릭하여 프로세스를 완료합니다.

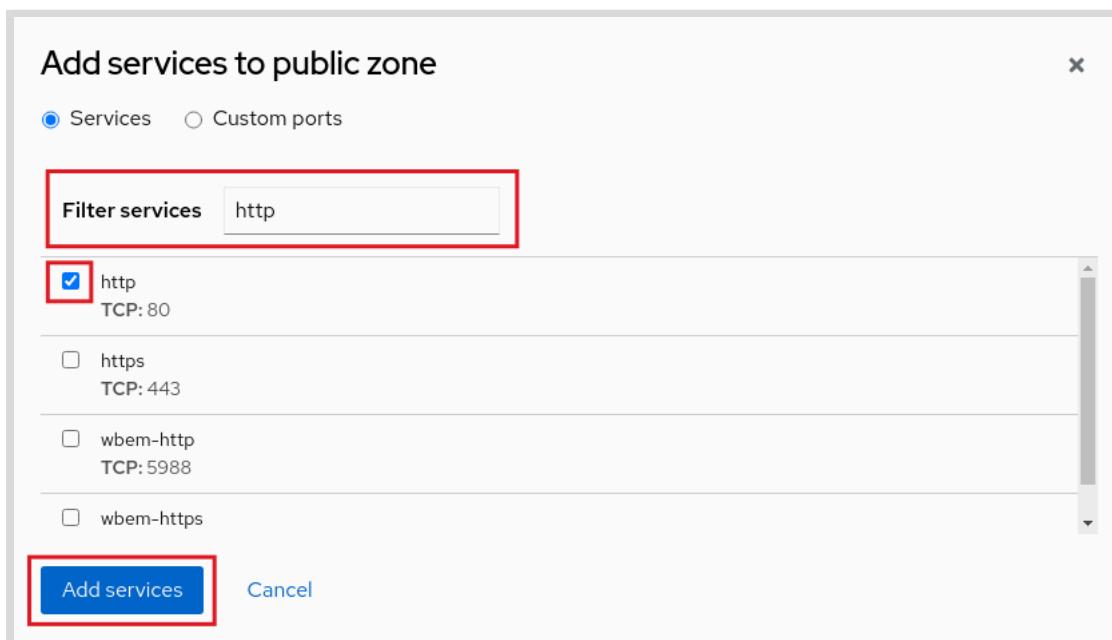


그림 15.4: 웹 콘솔에서 서비스 메뉴 옵션 추가

인터페이스가 **Firewall** 페이지로 돌아가고, 여기에서 업데이트된 허용된 서비스 목록을 검토할 수 있습니다.

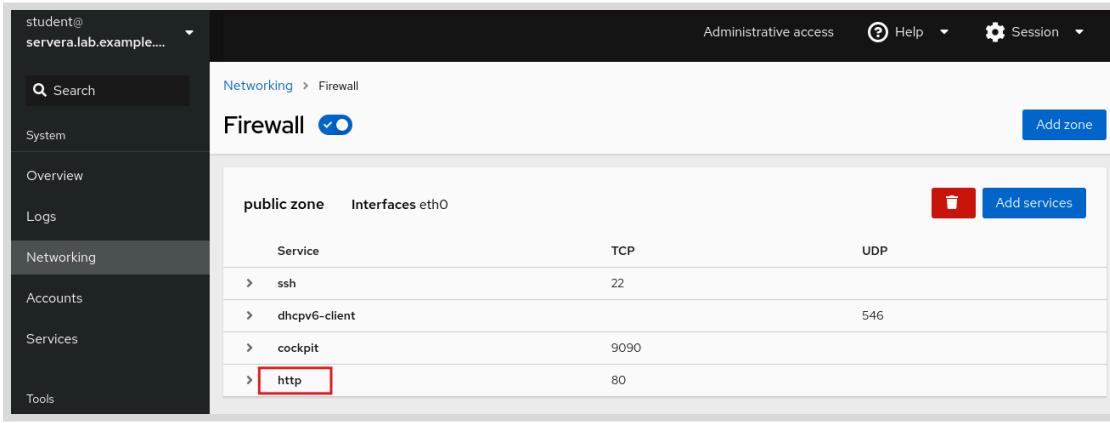


그림 15.5: 웹 콘솔 방화벽 개요

명령줄에서 방화벽 구성

`firewall-cmd` 명령은 `firewalld` 데몬과 인터페이스합니다. `firewalld` 패키지의 일부로 설치되며, 명령줄에서 작업하는 것을 선호하는 관리자를 위해 또는 그래픽 환경이 없는 시스템 작업이나 방화벽 설정 스크립트 작업을 위해 제공됩니다.

다음 표에는 자주 사용되는 `firewall-cmd` 명령과 설명이 나열되어 있습니다. `--permanent` 옵션을 지정하지 않는 한 대부분의 명령이 런타임 구성에서 실행됩니다. `--permanent` 옵션이 지정된 경우 `firewall-cmd --reload` 명령도 실행하여 해당 설정을 활성화해야 합니다. 이 명령은 현재의 영구 구성 읽고 이 구성을 새 런타임 구성으로 적용합니다. 나열된 명령 대부분은 `--zone=ZONE` 옵션을 사용하여 영향을 미치는 영역을 확인합니다. 넷마스크가 필요한 경우 CIDR 표기법(예: 192.168.1/24)을 사용합니다.

firewall-cmd 명령	설명
<code>--get-default-zone</code>	현재 기본 영역을 쿼리합니다.
<code>--set-default-zone=ZONE</code>	기본 영역을 설정합니다. 이 기본 영역은 런타임 및 영구 구성을 둘 다 변경합니다.
<code>--get-zones</code>	사용 가능한 모든 영역을 나열합니다.
<code>--get-active-zones</code>	현재 사용 중인 모든 영역(인터페이스가 있거나 연결된 소스가 있음)과 해당 인터페이스 및 소스 정보를 나열합니다.
<code>--add-source=CIDR [--zone=ZONE]</code>	IP 주소 또는 네트워크/넷마스크에서 오는 모든 트래픽을 지정된 영역으로 라우팅합니다. <code>--zone=</code> 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
<code>--remove-source=CIDR [--zone=ZONE]</code>	IP 주소 또는 네트워크에서 오는 모든 트래픽을 라우팅하는 규칙을 해당 영역에서 제거합니다. <code>--zone=</code> 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
<code>--add-interface=INTERFACE [--zone=ZONE]</code>	<code>INTERFACE</code> 의 모든 트래픽을 지정된 영역으로 라우팅합니다. <code>--zone=</code> 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.

firewall-cmd 명령	설명
--change-interface=INTERFACE [--zone=ZONE]	현재 영역 대신 인터페이스를 ZONE과 연결합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--list-all [--zone=ZONE]	ZONE에 대해 구성된 모든 인터페이스, 소스, 서비스, 포트를 나열합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--list-all-zones	모든 영역(인터페이스, 소스, 포트, 서비스)에 대한 모든 정보를 검색합니다.
--add-service=SERVICE [--zone=ZONE]	SERVICE에 대한 트래픽을 허용합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--add-port=PORT/PROTOCOL [--zone=ZONE]	PORT/PROTOCOL 포트에 대한 트래픽을 허용합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--remove-service=SERVICE [--zone=ZONE]	영역에 허용된 목록에서 SERVICE를 제거합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--remove-port=PORT/PROTOCOL [--zone=ZONE]	영역에 허용된 목록에서 PORT/PROTOCOL 포트를 제거합니다. --zone= 옵션을 입력하지 않은 경우 기본 영역이 사용됩니다.
--reload	런타임 구성을 삭제하고 영구 구성을 적용합니다.

아래 예제에서는 기본 영역을 **dmz**로 설정하고, **192.168.0.0/24** 네트워크에서 발생하는 모든 트래픽을 **internal** 영역에 할당하고, **mysql** 서비스의 네트워크 포트를 **internal** 영역에서 엽니다.

```
[root@host ~]# firewall-cmd --set-default-zone=dmz
[root@host ~]# firewall-cmd --permanent --zone=internal \
--add-source=192.168.0.0/24
[root@host ~]# firewall-cmd --permanent --zone=internal --add-service=mysql
[root@host ~]# firewall-cmd --reload
```

다른 예로, **172.25.25.11** 단일 IPv4 주소에서 들어오는 모든 트래픽을 **public** 영역에 추가하려면 다음 명령을 사용합니다.

```
[root@host ~]# firewall-cmd --permanent --zone=public \
--add-source=172.25.25.11/32
[root@host ~]# firewall-cmd --reload
```



참고

기본 구문이 충분하지 않은 경우, rich-rules 를 추가하여 복잡한 규칙을 작성할 수 있습니다. rich-rules 구문도 충분하지 않은 경우에는 직접 구성 규칙(**firewalld** 규칙과 혼합된 원시 **nft** 구문)을 사용할 수도 있습니다. 이러한 고급 구성은 이 장의 범위를 벗어납니다.



참조

`firewall-cmd(1)`, `firewalld(1)`, `firewalld.zone(5)`, `firewalld.zones(5)`,
`nft(8)` 도움말 페이지

▶ 연습 가이드

서버 방화벽 관리

이 연습에서는 **firewalld** 서비스로 시스템 방화벽 규칙을 조정하여 시스템 서비스에 대한 액세스를 제어합니다.

결과

- 서비스에 대한 액세스를 제어하는 방화벽 규칙을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start netsecurity-firewalls
```

지침

- ▶ 1. **servera** 시스템에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

- ▶ 2. **httpd** 및 **mod_ssl** 패키지를 설치합니다. 이러한 패키지는 Apache 웹 서버와 웹 서버가 SSL에서 콘텐츠를 지원하는데 필요한 확장 기능을 제공합니다.

```
[root@servera ~]# dnf install httpd mod_ssl
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- ▶ 3. **/var/www/html/index.html** 파일을 생성합니다. **I am servera.**라는 한 줄의 텍스트를 추가합니다.

```
[root@servera ~]# echo 'I am servera.' > /var/www/html/index.html
```

- ▶ 4. **httpd** 서비스를 시작하고 활성화합니다.

```
[root@servera ~]# systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/
lib/systemd/system/httpd.service.
```

▶ 5. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

▶ 6. workstation에서 일반 텍스트 포트 80/TCP 및 SSL 캡슐화 포트 443/TCP 를 둘 다 사용하여 servera 에서 웹 서버에 액세스합니다. 두 시도 모두 실패합니다.

6.1. curl 명령이 실패합니다.

```
[student@workstation ~]$ curl http://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 80: No route to host
```

6.2. 안전하지 않은 연결의 경우 curl 명령과 -k 옵션도 실패합니다.

```
[student@workstation ~]$ curl -k https://servera.lab.example.com
curl: (7) Failed to connect to servera.lab.example.com port 443: No route to host
```

▶ 7. servera 의 firewalld 서비스가 활성화되어 실행되고 있는지 확인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
  Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor
  preset: enabled)
  Active: active (running) since Wed 2022-04-13 11:22:50 EDT; 7min ago
    Docs: man:firewalld(1)
  Main PID: 768 (firewalld)
     Tasks: 2 (limit: 10798)
    Memory: 39.9M
      CPU: 584ms
     CGroup: /system.slice/firewalld.service
             └─768 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Apr 13 11:22:49 servera.lab.example.com systemd[1]: Starting firewalld - dynamic
firewall daemon...
Apr 13 11:22:50 servera.lab.example.com systemd[1]: Started firewalld - dynamic
firewall daemon.
```

▶ 8. https 서비스를 public 방화벽 영역에 추가합니다.

- 8.1. 기본 방화벽 영역이 public 영역으로 설정되었는지 확인합니다.

```
[root@servera ~]# firewall-cmd --get-default-zone
public
```

- 8.2. 위 단계에서 기본 영역으로 public이 반환되지 않은 경우 다음 명령을 사용하여 수정합니다.

```
[root@servera ~]# firewall-cmd --set-default-zone public
```

- 8.3. public 네트워크 영역의 영구 구성에 https 서비스를 추가합니다.構성을 확인합니다.

```
[root@servera ~]# firewall-cmd --permanent --add-service=https
success
[root@servera ~]# firewall-cmd --reload
success
[root@servera ~]# firewall-cmd --permanent --zone=public --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpcv6-client https ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

▶ 9. workstation에서 Firefox를 열고 servera에서 실행 중인 웹 콘솔에 로그인하여 public 방화벽 영역에 대한 https 서비스를 확인합니다.

- 9.1. Firefox를 열고 `https://servera.lab.example.com:9090` 으로 이동하여 웹 콘솔에 액세스합니다. Advanced 및 Accept the Risk and Continue를 클릭하여 자체 서명 인증서를 수락합니다.
- 9.2. student을 암호로 사용하여 student 사용자로 로그인합니다.
- 9.3. Turn on administrative access를 클릭하고 student 암호를 다시 입력합니다.
- 9.4. 왼쪽 네비게이션 바에서 Networking을 클릭합니다.
- 9.5. Networking 페이지의 Firewall 섹션에 있는 Edit rules and zones를 클릭합니다.
- 9.6. https 서비스가 Service 열에 나열되는지 확인합니다.

▶ 10. workstation의 터미널로 돌아가 servera 웹 서버에 액세스를 시도하여 작업을 확인합니다.

- 10.1. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

10.2. `http://servera.lab.example.com` 웹 서버에 대한 액세스를 확인합니다

```
[student@workstation ~]$ curl http://servera.lab.example.com  
curl: (7) Failed to connect to servera.lab.example.com port 80: No route to host
```

10.3. 비보안 연결을 위해 포트 443을 통해 `http://servera.lab.example.com` 웹 서버에 대한 액세스를 확인합니다.

```
[student@workstation ~]$ curl -k https://servera.lab.example.com  
I am servera.
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 위 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netsecurity-firewalls
```

이것으로 섹션을 완료합니다.

SELinux 포트 레이블 지정 제어

목표

네트워크 포트에 바인딩할 서비스에 맞는 올바른 SELinux 유형이 있는지 확인합니다.

SELinux 포트 레이블 지정

SELinux는 파일 컨텍스트 및 프로세스 유형 레이블 지정 외에도 SELinux 컨텍스트를 사용하여 네트워크 포트에 레이블을 지정합니다. SELinux는 네트워크 포트에 레이블을 지정하고 서비스의 타겟 정책에 규칙을 포함하여 네트워크 액세스를 제어합니다. 예를 들어 SSH 타겟 정책에는 `ssh_port_t` 포트 컨텍스트 레이블이 있는 `22/TCP` 포트가 포함됩니다. HTTP 정책에서 기본 `80/TCP` 및 `443/TCP` 포트는 `http_port_t` 포트 컨텍스트 레이블을 사용합니다.

타겟 프로세스에서 수신 대기를 위해 포트를 열려고 하면 프로세스와 컨텍스트를 바인딩할 수 있도록 SELinux가 정책에 항목이 포함되어 있는지 확인합니다. SELinux는 악성 서비스가 다른 합법적 네트워크 서비스에 사용되는 포트를 인계받지 못하도록 차단할 수 있습니다.

SELinux 포트 레이블 지정 관리

서비스가 비표준 포트에서 수신 대기를 시도하고 포트에 올바른 SELinux 유형의 레이블이 지정되지 않은 경우 SELinux에서 해당 시도를 차단할 수 있습니다. 포트에서 SELinux 컨텍스트를 변경하여 이 문제를 해결할 수 있습니다.

일반적으로 `targeted` 정책은 이미 예상되는 모든 포트에 올바른 유형으로 레이블을 지정했습니다. 예를 들어 웹 애플리케이션에서 `8008/TCP` 포트를 자주 사용하므로 해당 포트는 웹 서버에서 사용할 기본 포트 유형인 `http_port_t`로 레이블이 이미 지정되어 있습니다. 개별 포트는 하나의 포트 컨텍스트로만 레이블을 지정할 수 있습니다.

포트 레이블 나열

포트 번호를 필터링하려면 `grep` 명령을 사용합니다.

```
[root@host ~]# grep gopher /etc/services
gopher          70/tcp                                # Internet Gopher
gopher          70/udp
```

현재 포트 레이블 할당을 나열하려면 `semanage` 명령을 사용합니다.

```
[root@host ~]# semanage port -l
...output omitted...
http_cache_port_t      tcp    8080, 8118, 8123, 10001-10010
http_cache_port_t      udp    3130
http_port_t            tcp    80, 81, 443, 488, 8008, 8009, 8443, 9000
...output omitted...
```

서비스 이름을 사용하여 SELinux 포트 레이블을 필터링하려면 `grep` 명령을 사용합니다.

```
[root@host ~]# semanage port -l | grep ftp
ftp_data_port_t          tcp      20
ftp_port_t                tcp      21, 989, 990
ftp_port_t                udp      989, 990
tftp_port_t               udp      69
```

포트 레이블은 지원되는 각 네트워킹 프로토콜에 대해 목록에 여러 번 표시될 수 있습니다.

포트 번호를 사용하여 SELinux 포트 레이블을 필터링하려면 **grep** 명령을 사용합니다.

```
[root@host ~]# semanage port -l | grep -w 70
gopher_port_t             tcp      70
gopher_port_t             udp      70
```

포트 바인딩 관리

새 포트 레이블을 할당하거나, 포트 레이블을 제거하거나, 기존 포트 레이블을 수정하려면 **semanage** 명령을 사용합니다.



중요

RHEL 배포에 포함된 거의 모든 서비스에서는 해당 서비스의 기본 포트 컨텍스트를 포함하는 SELinux 정책 모듈을 제공합니다. **semanage** 명령으로는 기본 포트 레이블을 변경할 수 없습니다. 대신 타겟 서비스의 정책 모듈을 수정하고 다시 로드해야 합니다. 정책 모듈 작성 및 생성 작업은 이 교육 과정에서 설명하지 않습니다.

기존 포트 컨텍스트 레이블(유형)을 사용하여 새 포트에 레이블을 지정할 수 있습니다. **semanage port** 명령의 **-a** 옵션은 새 포트 레이블을 추가하고, **-t** 옵션을 유형을, **-p** 옵션은 프로토콜을 나타냅니다.

```
[root@host ~]# semanage port -a -t port_label -p tcp|udp PORTNUMBER
```

다음 예제에서는 **gopher** 서비스를 활성화하여 71/TCP 포트에서 수신 대기합니다.

```
[root@host~]# semanage port -a -t gopher_port_t -p tcp 71
```

기본 정책에 대한 로컬 변경 사항을 보려면 **-C** 명령의 **semanage port** 옵션을 사용합니다.

```
[root@host~]# semanage port -l -C
SELinux Port Type           Proto   Port Number
gopher_port_t                tcp     71
```

타겟 정책에는 다양한 포트 유형이 포함됩니다.

서비스별 SELinux 도움말 페이지는 서비스 이름과 **_selinux**를 사용하여 이름이 지정됩니다. 이러한 도움말 페이지에는 SELinux 유형, 부울, 포트 유형에 대한 서비스별 정보가 포함되며 기본적으로 설치되지는 않습니다. 사용 가능한 모든 SELinux 도움말 페이지 목록을 보려면 패키지를 설치한 다음 **_selinux** 문자열에 대해 **man -k** 키워드 검색을 실행하십시오.

```
[root@host ~]# dnf -y install selinux-policy-doc
[root@host ~]# man -k _selinux
```

-d 옵션과 함께 포트 레이블을 삭제하려면 **semanage** 명령을 사용하십시오. 다음 예제에서 포트 71/TCP에서 **gopher_port_t** 유형으로의 바인딩을 제거합니다.

```
[root@host ~]# semanage port -d -t gopher_port_t -p tcp 71
```

포트 바인딩을 변경하려면 요구 사항이 변경될 때 **-m** 옵션을 사용합니다. 이 옵션이 이전 바인딩을 삭제한 후 새 바인딩을 추가하는 것보다 더 효율적입니다.

예를 들어 포트 71/TCP를 **gopher_port_t**에서 **http_port_t**로 수정하려면 다음 명령을 사용합니다.

```
[root@server ~]# semanage port -m -t http_port_t -p tcp 71
```

semanage 명령을 사용하여 수정 사항을 봅니다.

```
[root@server ~]# semanage port -l -c
SELinux Port Type          Proto    Port Number

http_port_t                tcp      71
[root@server ~]# semanage port -l | grep http
http_cache_port_t           tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t           udp      3130
http_port_t                tcp      71, 80, 81, 443, 488, 8008, 8009, 8443,
                                9000
pegasus_http_port_t         tcp      5988
pegasus_https_port_t        tcp      5989
```



참조

[semanage\(8\)](#), [semanage-port\(8\)](#) 및 [*_selinux\(8\)](#) 도움말 페이지

▶ 연습 가이드

SELinux 포트 레이블 지정 제어

이 랩에서는 비표준 포트에서 HTTP 액세스를 허용하도록 시스템을 구성합니다.

결과

- servera에서 실행되며 비표준 포트를 사용하여 콘텐츠를 성공적으로 제공하는 웹 서버를 구성합니다.

시작하기 전에

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 네트워크에서 servera 시스템에 연결할 수 있는지를 확인하고, httpd 서비스를 설치하고, HTTP 연결을 허용하도록 servera에 방화벽을 구성합니다.

```
[student@workstation ~]$ lab start netsecurity-ports
```

지침

조직에서는 새로운 사용자 지정 웹 애플리케이션을 배포하고 있습니다. 웹 애플리케이션은 비표준 포트(이 경우 82/TCP)에서 실행됩니다.

하위 관리자 한 명이 servera 호스트에 이미 애플리케이션을 구성했습니다. 그러나 웹 서버 콘텐츠에는 액세스할 수 없습니다.

▶ 1. servera에 student 사용자로 로그인한 후 root 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

▶ 2. httpd 서비스를 다시 시작하여 웹 콘텐츠 문제를 해결합니다.

2.1. httpd.service를 다시 시작합니다. 이 명령은 실패할 것으로 예상됩니다.

```
[root@servera ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xe" for details.
```

2.2. httpd 서비스의 상태를 봅니다. permission denied 오류를 확인합니다.

```
[root@servera ~]# systemctl status -l httpd.service
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: failed (Result: exit-code) since Mon 2019-04-08 14:23:29 CEST; 3min 33s ago
     Docs: man:httpd.service(8)
   Process: 28078 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
 Main PID: 28078 (code=exited, status=1/FAILURE)
    Status: "Reading configuration..."

Apr 08 14:23:29 servera.lab.example.com systemd[1]: Starting The Apache HTTP Server...
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission denied:
AH00072: make_sock: could not bind to address [::]:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: (13)Permission denied:
AH00072: make_sock: could not bind to address 0.0.0.0:82
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: no listening sockets available, shutting down
Apr 08 14:23:29 servera.lab.example.com httpd[28078]: AH00015: Unable to open logs
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service: Main process exited, code=exited, status=1/FAILURE
Apr 08 14:23:29 servera.lab.example.com systemd[1]: httpd.service: Failed with result 'exit-code'.
Apr 08 14:23:29 servera.lab.example.com systemd[1]: Failed to start The Apache HTTP Server.
```

2.3. SELinux로 인해 httpd 가 82/TCP 포트에 바인딩하지 못하는지 확인합니다.

```
[root@servera ~]# sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
-----
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port 82.

***** Plugin bind_ports (99.5 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 82
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 82 where PORT_TYPE is one of the following:
http_cache_port_t, http_port_t, jboss_management_port_t, jboss.messaging_port_t,
ntop_port_t, puppet_port_t.
...output omitted...
Raw Audit Messages
type=AVC msg=audit(1554726569.188:852): avc: denied { name_bind } for
pid=28393 comm="httpd" src=82 scontext=system_u:system_r:httpd_t:s0
tcontext=system_u:object_r:reserved_port_t:s0 tclass=tcp_socket permissive=0
...output omitted...
```

▶ 3. **httpd** 서비스가 82/TCP 포트에 바인딩할 수 있게 SELinux를 구성하고 **httpd.service** 서비스를 다시 시작합니다.

3.1. 82/TCP 포트의 적절한 포트 유형을 찾습니다.

http_port_t 유형에는 기본 HTTP 포트인 80/TCP 및 443/TCP가 포함됩니다. 이것이 웹 서버의 올바른 포트 유형입니다.

```
[root@servera ~]# semanage port -l | grep http
http_cache_port_t          tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t          udp      3130
http_port_t                tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t        tcp      5988
pegasus_https_port_t       tcp      5989
```

3.2. 82/TCP 포트에 **http_port_t** 유형을 할당합니다.

```
[root@servera ~]# semanage port -a -t http_port_t -p tcp 82
```

3.3. **httpd.service** 서비스를 다시 시작합니다. 이 명령은 성공합니다.

```
[root@servera ~]# systemctl restart httpd.service
```

▶ 4. 이제 82/TCP 포트에서 실행되는 웹 서버에 액세스할 수 있는지 확인합니다.

```
[root@servera ~]# curl http://servera.lab.example.com:82
Hello
```

▶ 5. 다른 터미널 창을 사용하여 **workstation**에서 새 웹 서비스에 액세스할 수 있는지 확인합니다.

```
[student@workstation ~]$ curl http://servera.lab.example.com:82
curl: (7) Failed to connect to servera.example.com:82; No route to host
```

이 오류는 여전히 **workstation**에서 웹 서비스에 연결할 수 없음을 나타냅니다.

▶ 6. **servera**에서 방화벽의 82/TCP 포트를 엽니다.

6.1. **servera**의 방화벽에서 기본 영역에 대한 영구 구성의 82/TCP 포트를 엽니다.

```
[root@servera ~]# firewall-cmd --permanent --add-port=82/tcp
success
```

6.2. **servera**에서 방화벽 변경 내용을 활성화합니다.

```
[root@servera ~]# firewall-cmd --reload
success
```

▶ 7. **workstation**에서 웹 서비스에 액세스합니다.

```
[student@workstation ~]$ curl http://servera.lab.example.com:82  
Hello
```

▶ 8. workstation 시스템에 student 사용자로 돌아갑니다.

```
[root@servera ~]# exit  
logout  
[student@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netsecurity-ports
```

이것으로 섹션을 완료합니다.

▶ 랩

네트워크 보안 관리

이 랩에서는 방화벽 및 SELinux 설정을 구성하여 동일한 호스트에서 실행되는 여러 웹 서버에 대한 액세스를 허용합니다.

결과

- 웹 서버 호스트에서 방화벽 및 SELinux 설정을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start netsecurity-review
```

지침

귀사에서 새 웹 애플리케이션을 실행하기로 했습니다. 애플리케이션은 **80/TCP** 및 **1001/TCP** 포트에서 수신 대기합니다. 모든 변경 사항은 재부팅 때마다 유지되어야 합니다.



중요

Red Hat 온라인 학습 환경에서는 **5900/TCP** 포트에서 그래픽 인터페이스를 계속 사용할 수 있어야 합니다. 이 포트는 **vnc-server** 서비스로도 알려져 있습니다. **serverb** 시스템에서 사용자 자신을 실수로 잠근 경우 **workstation** 시스템에서 **serverb** 시스템에 대해 **ssh** 명령을 사용하여 복구를 시도하거나 **serverb** 시스템을 재설정하면 됩니다. **serverb** 시스템을 재설정할 경우 이 랩에서 설정 스크립트를 다시 실행해야 합니다. 시스템 구성에 이미 이러한 포트를 여는 **ROL**이라는 사용자 지정 영역이 포함되어 있습니다.

1. **workstation** 시스템에서 **http://serverb.lab.example.com** 및 **http://serverb.lab.example.com:1001** 가상 호스트를 봅니다.
2. **serverb** 시스템에 로그인하여 웹 서버에 액세스할 수 없는 이유를 확인합니다.
3. **httpd** 서비스가 **1001/TCP** 포트에서 수신 대기하도록 SELinux를 구성합니다.
4. **workstation** 시스템에서 **http://serverb.lab.example.com** 및 **http://serverb.lab.example.com:1001** 가상 호스트를 봅니다.
5. **serverb** 시스템에 로그인하여 방화벽에 올바른 포트가 할당되어 있는지 확인합니다.
6. **public** 네트워크 영역의 영구 구성에 **1001/TCP** 포트를 추가합니다. 구성을 확인합니다.
7. **workstation**에서 **http://serverb.lab.example.com**에 있는 기본 웹 서버는 **SERVER B**를 반환하고, **http://serverb.lab.example.com:1001**에 있는 가상 호스트는 **VHOST 1**을 반환하는지 확인합니다.

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade netsecurity-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netsecurity-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

네트워크 보안 관리

이 랩에서는 방화벽 및 SELinux 설정을 구성하여 동일한 호스트에서 실행되는 여러 웹 서버에 대한 액세스를 허용합니다.

결과

- 웹 서버 호스트에서 방화벽 및 SELinux 설정을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start netsecurity-review
```

지침

귀사에서 새 웹 애플리케이션을 실행하기로 했습니다. 애플리케이션은 **80/TCP** 및 **1001/TCP** 포트에서 수신 대기합니다. 모든 변경 사항은 재부팅 때마다 유지되어야 합니다.



중요

Red Hat 온라인 학습 환경에서는 **5900/TCP** 포트에서 그래픽 인터페이스를 계속 사용할 수 있어야 합니다. 이 포트는 **vnc-server** 서비스로도 알려져 있습니다. **serverb** 시스템에서 사용자 자신을 실수로 잠근 경우 **workstation** 시스템에서 **serverb** 시스템에 대해 **ssh** 명령을 사용하여 복구를 시도하거나 **serverb** 시스템을 재설정하면 됩니다. **serverb** 시스템을 재설정할 경우 이 랩에서 설정 스크립트를 다시 실행해야 합니다. 시스템 구성에 이미 이러한 포트를 여는 **ROL**이라는 사용자 지정 영역이 포함되어 있습니다.

1. **workstation** 시스템에서 **http://serverb.lab.example.com** 및 **http://serverb.lab.example.com:1001** 가상 호스트를 봅니다.
 - 1.1. **http://serverb.lab.example.com** 웹 서버에 대한 액세스를 테스트합니다 현재 테스트에 실패했습니다. 웹 서버에서 **SERVER B**를 반환해야 합니다.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
curl: (7) Failed to connect to serverb.lab.example.com port 80: Connection refused
```

- 1.2. **http://serverb.lab.example.com:1001** 가상 호스트에 대한 액세스 권한을 테스트합니다. 현재 테스트에 실패했습니다. 가상 호스트에서 **VHOST 1**을 반환해야 합니다.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

2. **serverb** 시스템에 로그인하여 웹 서버에 액세스할 수 없는 이유를 확인합니다.

2.1. **student** 사용자로 **serverb** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2.2. **httpd** 서비스가 활성 상태인지 확인합니다.

```
[student@serverb ~]$ systemctl is-active httpd
inactive
```

2.3. **httpd** 서비스를 활성화하고 시작합니다. **httpd** 서버가 시작되지 않습니다.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
[sudo] password for student: student
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/
lib/systemd/system/httpd.service.
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
```

2.4. **httpd** 서비스가 시작되지 못한 이유를 조사합니다.

```
[student@serverb ~]$ systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor
preset: disabled)
     Active: failed (Result: exit-code) since Wed 2022-04-13 06:55:01 EDT; 2min
52s ago
       Docs: man:httpd.service(8)
      Process: 1640 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
status=1/FAILURE)
     Main PID: 1640 (code=exited, status=1/FAILURE)
        Status: "Reading configuration..."
          CPU: 31ms

Apr 13 06:55:01 serverb.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: (13)Permission denied:
AH00072: make_sock: could not bind to address [::]:1001
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: (13)Permission denied:
AH00072: make_sock: could not bind to address 0.0.0.0:1001
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: no listening sockets
available, shutting down
Apr 13 06:55:01 serverb.lab.example.com httpd[1640]: AH00015: Unable to open logs
```

```
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: httpd.service: Main process
exited, code=exited, status=1/FAILURE
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: httpd.service: Failed with
result 'exit-code'.
Apr 13 06:55:01 serverb.lab.example.com systemd[1]: Failed to start The Apache
HTTP Server.
```

2.5. SELinux로 인해 **httpd** 서비스가 **1001/TCP** 포트에 바인딩하지 못하는지 확인합니다.

```
[student@serverb ~]$ sudo sealert -a /var/log/audit/audit.log
100% done
found 1 alerts in /var/log/audit/audit.log
-----
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port
1001.

***** Plugin bind_ports (99.5 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 1001
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 1001
    where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,
jboss_management_port_t, jboss.messaging_port_t, ntop_port_t, puppet_port_t.

***** Plugin catchall (1.49 confidence) suggests *****
...output omitted...
```

3. **httpd** 서비스가 **1001/TCP** 포트에서 수신 대기하도록 SELinux를 구성합니다.

3.1. **semanage** 명령을 사용하여 올바른 포트 유형을 찾습니다.

```
[student@serverb ~]$ sudo semanage port -l | grep 'http'
http_cache_port_t          tcp      8080, 8118, 8123, 10001-10010
http_cache_port_t          udp      3130
http_port_t                tcp      80, 81, 443, 488, 8008, 8009, 8443, 9000
pegasus_http_port_t         tcp      5988
pegasus_https_port_t        tcp      5989
```

3.2. **1001/TCP** 포트를 **http_port_t** 유형에 바인딩합니다.

```
[student@serverb ~]$ sudo semanage port -a -t http_port_t -p tcp 1001
```

3.3. **1001/TCP** 포트가 **http_port_t** 포트 유형에 바인딩되어 있는지 확인합니다.

```
[student@serverb ~]$ sudo semanage port -l | grep '^http_port_t'
http_port_t              tcp      1001, 80, 81, 443, 488, 8008, 8009, 8443, 9000
```

3.4. **httpd** 서비스를 활성화하고 시작합니다.

```
[student@serverb ~]$ sudo systemctl enable --now httpd
```

3.5. **httpd** 서비스의 실행 상태를 확인합니다.

```
[student@serverb ~]$ systemctl is-active httpd
active
[student@serverb ~]$ systemctl is-enabled httpd
enabled
```

3.6. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

4. **workstation** 시스템에서 **http://serverb.lab.example.com** 및 **http://serverb.lab.example.com:1001** 가상 호스트를 봅니다.

4.1. **http://serverb.lab.example.com** 웹 서버에 대한 액세스를 테스트합니다 웹 서버에서 SERVER B를 반환해야 합니다.

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

4.2. **http://serverb.lab.example.com:1001** 가상 호스트에 대한 액세스 권한을 테스트합니다. 테스트가 계속 실패합니다.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host
```

5. **serverb** 시스템에 로그인하여 방화벽에 올바른 포트가 할당되어 있는지 확인합니다.

5.1. **student** 사용자로 **serverb** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

5.2. 기본 방화벽 영역이 **public** 영역으로 설정되었는지 확인합니다.

```
[student@serverb ~]$ firewall-cmd --get-default-zone
public
```

5.3. 위 단계에서 기본 영역으로 **public** 이 반환되지 않은 경우 다음 명령을 사용하여 수정합니다.

```
[student@serverb ~]$ sudo firewall-cmd --set-default-zone public
```

5.4. **public** 네트워크 영역에 나열된 열려 있는 포트를 확인합니다.

```
[student@serverb ~]$ sudo firewall-cmd --zone=public --list-all
[sudo] password for student: student
public
```

```
target: default
icmp-block-inversion: no
interfaces:
sources:
services: cockpit dhcpcv6-client http ssh
ports:
protocols:
forward: yes
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

6. **public** 네트워크 영역의 영구 구성에 **1001/TCP** 포트를 추가합니다. 구성을 확인합니다.

6.1. **1001/TCP** 포트를 **public** 네트워크 영역에 추가합니다.

```
[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \
--add-port=1001/tcp
success
```

6.2. 방화벽 구성을 다시 로드합니다.

```
[student@serverb ~]$ sudo firewall-cmd --reload
success
```

6.3. 구성을 확인합니다.

```
[student@serverb ~]$ sudo firewall-cmd --zone=public --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: cockpit dhcpcv6-client http ssh
  ports: 1001/tcp
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

6.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

7. workstation에서 `http://serverb.lab.example.com`에 있는 기본 웹 서버는 SERVER B를 반환하고, `http://serverb.lab.example.com:1001`에 있는 가상 호스트는 VHOST 1을 반환하는지 확인합니다.

7.1. `http://serverb.lab.example.com` 웹 서버에 대한 액세스를 테스트합니다

```
[student@workstation ~]$ curl http://serverb.lab.example.com
SERVER B
```

7.2. `http://serverb.lab.example.com:1001` 가상 호스트에 대한 액세스 권한을 테스트합니다.

```
[student@workstation ~]$ curl http://serverb.lab.example.com:1001
VHOST 1
```

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade netsecurity-review
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish netsecurity-review
```

이것으로 섹션을 완료합니다.

요약

- **netfilter** 프레임워크에서는 kernel 모듈이 들어오거나 나가거나 전달되는 모든 네트워크 패킷을 포함하여 시스템을 통과하는 패킷을 모두 검사할 수 있습니다.
- **firewalld** 서비스는 모든 네트워크 트래픽을 영역으로 분류하여 관리를 간소화합니다. 각 영역에는 포트 및 서비스에 대한 자체 목록이 있습니다. **public** 영역은 기본 영역으로 설정되어 있습니다.
- **firewalld** 서비스에는 사전 정의 서비스가 제공됩니다. 4288 **firewall-cmd --get-services** 명령을 사용하여 해당 서비스를 나열할 수 있습니다.
- SELinux 정책은 네트워크 포트에 레이블을 지정하여 네트워크 트래픽을 제어합니다. 예를 들어 **22/TCP** 포트에는 **ssh_port_t** 레이블이 연결되어 있습니다. 프로세스가 포트에서 수신 대기하려고 하면 SELinux는 연결된 레이블이 해당 포트 레이블에 바인딩될 수 있는지 확인합니다.
- **semanage** 명령은 레이블을 추가, 삭제, 수정하는 데 사용됩니다.

컨테이너 실행

목적

단일 Red Hat Enterprise Linux Server에서 간단한 경량 서비스를 컨테이너 형태로 가져와 실행하고 관리합니다.

목표

- 컨테이너의 개념과 컨테이너를 빌드, 저장, 실행하는 데 필요한 핵심 기술을 설명합니다.
- 레지스트리를 사용하여 이미지를 저장 및 검색하고, 컨테이너를 배포 및 쿼리하고 컨테이너에 액세스하는 컨테이너 관리 툴을 설명합니다.
- 컨테이너 호스트에서 스토리지를 공유하여 컨테이너 데이터를 위한 영구저장장치를 제공하고 컨테이너 네트워크를 구성합니다.
- 컨테이너를 **systemd** 서비스로 구성하고 부팅 시 시작되도록 컨테이너 서비스를 구성합니다.

섹션

- 컨테이너 개념(퀴즈 포함)
- 컨테이너 배포(안내에 따른 연습)
- 컨테이너 스토리지 및 네트워크 리소스 관리(안내에 따른 연습)
- 컨테이너를 시스템 서비스로 관리(안내에 따른 연습)

랩

컨테이너 실행

컨테이너 개념

목표

컨테이너의 개념과 컨테이너를 빌드, 저장, 실행하는 데 필요한 핵심 기술을 설명합니다.

컨테이너 기술

소프트웨어 애플리케이션은 일반적으로 런타임 환경에서 제공하는 라이브러리, 구성 파일 또는 서비스에 의존합니다. 소프트웨어 애플리케이션의 런타임 환경은 보통 물리적 호스트 또는 가상 시스템에서 실행되는 운영 체제에 설치됩니다. 그러면 관리자가 운영 체제 위에 애플리케이션 종속성을 설치합니다.

Red Hat Enterprise Linux에서 RPM과 같은 패키징 시스템은 관리자가 애플리케이션 종속성을 관리하는데 도움이 됩니다. RPM 시스템은 `httpd` 패키지를 설치할 때 해당 패키지의 올바른 라이브러리 및 기타 종속성이 함께 설치되도록 합니다.

기존에 배포된 소프트웨어 애플리케이션의 주요 단점은 이러한 종속성이 런타임 환경과 얹혀 있다는 점입니다. 애플리케이션에는 운영 체제와 함께 제공된 지원 소프트웨어보다 이전 또는 이후 버전의 지원 소프트웨어가 필요할 수 있습니다. 마찬가지로, 동일한 시스템에 있는 두 개의 애플리케이션에서 동일한 소프트웨어를 서로 호환되지 않는 다른 버전으로 요구할 수도 있습니다.

이러한 충돌을 해결하는 한 가지 방법은 애플리케이션을 컨테이너로 패키징하여 배포하는 것입니다. 컨테이너는 나머지 시스템에서 격리된 하나 이상의 프로세스 집합입니다. 소프트웨어 컨테이너는 애플리케이션을 패키징하고 배포 및 관리를 단순화하는 방법을 제공합니다.

물리적인 배송 컨테이너를 생각해 보십시오. 배송 컨테이너는 상품을 포장하고 배송하는 기본적인 수단입니다. 배송 컨테이너에 레이블을 붙이고, 짐을싣고, 내리고, 컨테이너 전체를 한 장소에서 다른 장소로 운송합니다. 컨테이너의 내용물은 다른 컨테이너의 내용물과 분리되므로 서로 영향을 주지 않습니다. 이러한 기본 원칙은 소프트웨어 컨테이너에도 적용됩니다.

Red Hat Enterprise Linux는 다음과 같은 핵심 기술을 사용하여 컨테이너를 지원합니다.

- 리소스 관리를 위한 cGroups(제어 그룹)
- 프로세스 분리를 위한 Namespaces
- 보안 경계를 강제 적용하는 SELinux 및 Seccomp (보안 컴퓨팅 모드)



참고

컨테이너 아키텍처 및 보안에 관한 자세한 내용은 "컨테이너 보안의 10개 계층" [<https://www.redhat.com/en/resources/container-security-openshift-cloud-devops-whitepaper>] 백서를 참조하십시오.

컨테이너와 가상 시스템의 차이점

컨테이너는 보안, 스토리지 및 네트워크 격리와 같은 가상 시스템과 동일한 이점을 많이 제공합니다.

두 기술 모두 호스트 운영 체제 또는 하이퍼바이저에서 애플리케이션 라이브러리와 런타임 리소스를 분리하거나 그 반대로 분리합니다.

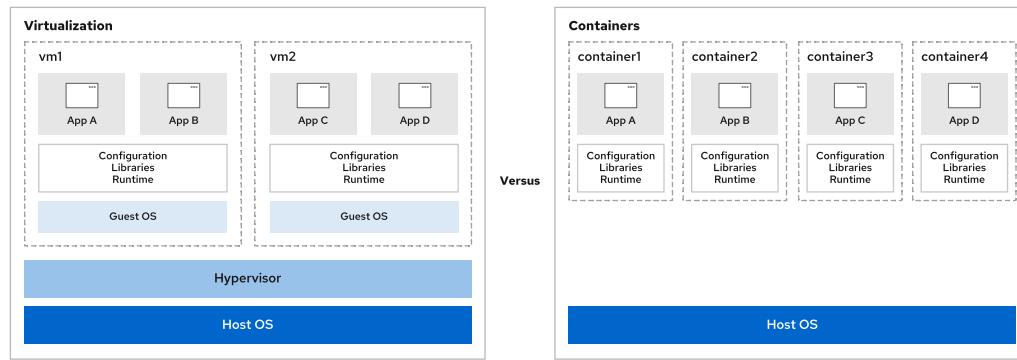


그림 16.1: 가상화와 컨테이너화 비교

컨테이너와 가상 시스템은 하드웨어 및 기본 운영 체제와 상호 작용하는 방식이 다릅니다.

가상 시스템에는 다음과 같은 특징이 있습니다.

- 단일 하드웨어 플랫폼에서 동시에 여러 운영 체제를 실행할 수 있도록 합니다.
- 하이퍼바이저를 사용하여 하드웨어를 여러 개의 가상 하드웨어 시스템으로 나눕니다.
- 전체 운영 체제 환경에서 해당 애플리케이션을 지원해야 합니다.

컨테이너에는 다음과 같은 특징이 있습니다.

- 운영 체제에서 직접 실행되며 시스템의 모든 컨테이너와 리소스를 공유합니다.
- 호스트의 커널을 공유하지만 시스템의 나머지 부분과 애플리케이션 프로세스를 격리합니다.
- 가상 시스템보다 필요한 하드웨어 리소스가 훨씬 적어 컨테이너를 더 빠르게 시작할 수 있습니다.
- 시스템 및 프로그래밍 종속성, 구성 설정 등의 모든 종속성을 포함합니다.



참고

컨테이너로 실행하기에 적합하지 않은 애플리케이션도 있습니다. 예를 들어 낮은 수준의 하드웨어 정보에 액세스하는 애플리케이션에는 컨테이너에서 일반적으로 제공하는 액세스 권한보다 더 직접적인 하드웨어 액세스가 필요할 수 있습니다.

Rootless 및 Rootful 컨테이너

컨테이너 호스트에서 root 사용자 또는 권한이 없는 일반 사용자로 컨테이너를 실행할 수 있습니다. 권한 있는 사용자가 실행하는 컨테이너를 rootful 컨테이너라고 합니다. 권한 없는 사용자가 실행하는 컨테이너를 rootless 컨테이너라고 합니다.

rootless 컨테이너는 제한된 디렉터리에 대한 액세스와 같이 일반적으로 권한 있는 사용자를 위해 예약된 시스템 리소스를 사용하거나 제한된 포트(1024 미만)에 네트워크 서비스를 게시할 수 없습니다. 이 기능으로 인해 잠재적인 공격자는 컨테이너 호스트에 대한 루트 권한을 얻을 수 없습니다.

필요에 따라 **root**로 직접 컨테이너를 실행할 수 있지만, 그러면 시스템의 보안이 저하되고 공격자가 버그를 이용하여 컨테이너를 손상시킬 수 있습니다.

컨테이너 기반 아키텍처 설계

컨테이너는 효율적으로 호스팅된 애플리케이션을 재사용하고 이식합니다. 컨테이너는 예를 들어 개발 환경에서 프로덕션 환경으로 이동할 수 있으며, 여러 버전의 컨테이너를 저장해 두고 필요에 따라 각 버전에 액세스할 수 있습니다.

컨테이너는 대개 임시로 사용하거나 사용 후 삭제됩니다. 실행 중인 컨테이너에서 생성하는 데이터는 영구저장장치에 영구적으로 저장할 수 있지만, 컨테이너 자체는 일반적으로 필요할 때 실행한 다음 중지 후 제거합니다. 다음번에 그 컨테이너가 필요하면 새 컨테이너 프로세스를 시작합니다.

다양한 서비스가 포함된 복잡한 소프트웨어 애플리케이션을 단일 컨테이너에 설치할 수 있습니다. 예를 들어 웹 서버에서 데이터베이스와 메시징 시스템을 사용해야 할 수 있습니다. 그러나 여러 서비스에 하나의 컨테이너를 사용하면 관리하기가 어렵습니다.

각각의 구성 요소, 웹 서버, 데이터베이스, 메시징 시스템을 개별 컨테이너에서 실행하는 것이 더 나은 설계입니다. 이렇게 하면 개별 애플리케이션 구성 요소를 업데이트 및 유지 관리해도 다른 구성 요소 또는 애플리케이션 스택에 영향을 주지 않습니다.

컨테이너 관리 툴

Red Hat Enterprise Linux에서는 단일 서버에서 여러 컨테이너를 실행하는 데 사용할 수 있는 컨테이너 툴 집합을 제공합니다.

- **podman**은 컨테이너 및 컨테이너 이미지를 관리합니다.
- **skopeo**는 이미지를 검사, 복사, 삭제하고 이미지에 서명합니다.
- **buildah**는 컨테이너 이미지를 생성합니다.

이러한 툴은 OCI(Open Container Initiative)와 호환되며, 이러한 툴을 사용하면 Podman 또는 Docker와 같은 OCI 호환 컨테이너 엔진에서 생성하는 모든 Linux 컨테이너를 관리할 수 있습니다. 해당 툴은 단일 노드 컨테이너 호스트의 Red Hat Enterprise Linux에서 컨테이너를 실행하도록 설계되었습니다.

이 장에서는 **podman** 및 **skopeo** 유ти리티를 사용하여 컨테이너와 기존 컨테이너 이미지를 실행하고 관리합니다.



참고

buildah를 사용하여 고유한 컨테이너 이미지를 구성하는 것은 이 교육 과정의 범위를 벗어납니다. 이 내용은 Red Hat OpenShift I: Containers & Kubernetes (DO180) Red Hat 교육 과정에서 다룹니다.

컨테이너 이미지 및 레지스트리

컨테이너를 실행하려면 컨테이너 이미지를 사용해야 합니다. 컨테이너 이미지는 코드화된 단계를 포함하는 정적 파일이며 컨테이너를 생성하는 청사진 역할을 합니다. 컨테이너 이미지는 시스템 라이브러리, 프로그래밍 언어 런타임 및 라이브러리, 기타 구성 설정 등의 모든 종속성과 함께 애플리케이션을 패키징합니다.

컨테이너 이미지는 OCI(Open Container Initiative) 이미지 형식 사양과 같은 사양에 따라 빌드됩니다. 이러한 사양에서는 컨테이너 이미지의 형식과 해당 이미지를 사용할 수 있는 컨테이너 호스트 운영 체제 및 하드웨어 아키텍처에 관한 메타데이터를 정의합니다.

컨테이너 레지스트리는 컨테이너 이미지를 저장하고 검색할 수 있는 리포지토리입니다. 개발자가 컨테이너 이미지를 컨테이너 레지스트리로 내보내거나 업로드합니다. 사용자는 컨테이너를 실행할 컨테이너 이미지를 레지스트리에서 로컬 시스템으로 가져오거나 다운로드합니다.

타사 이미지가 포함된 퍼블릭 레지스트리를 사용하거나 사용자 조직에서 관리하는 프라이빗 레지스트리를 사용할 수 있습니다. 중요한 것은 컨테이너 이미지의 소스입니다. 다른 소프트웨어 패키지와 마찬가지로 컨테이너 이미지의 코드를 신뢰할 수 있는지 확인해야 합니다. 제출된 컨테이너 이미지의 제공, 평가, 테스트 여부와 그 방법에 관한 정책은 레지스트리마다 서로 다릅니다.

Red Hat에서는 Red Hat 로그인 자격 증명을 사용하여 액세스할 수 있는 두 개의 기본 컨테이너 레지스트리를 통해 인증된 컨테이너 이미지를 배포합니다.

- **registry.redhat.io**: 공식 Red Hat 제품을 기반으로 하는 컨테이너용
- **registry.connect.redhat.com**: 타사 제품을 기반으로 하는 컨테이너용

Red Hat Container Catalog(<https://access.redhat.com/containers>)는 이러한 레지스트리에서 인증된 콘텐츠를 검색할 수 있는 웹 기반 인터페이스를 제공합니다.



참고

Red Hat은 컨테이너를 빌드하기 위한 초기 계층으로 UBI(Universal Base Image) 이미지를 제공합니다. UBI 이미지는 애플리케이션 빌드의 첫 번째 계층이 될 수 있는 최소화된 컨테이너 이미지입니다.

Red Hat 레지스트리에서 이미지를 다운로드하려면 Red Hat Developer 계정이 필요합니다. **podman login** 명령을 사용하여 레지스트리에 인증할 수 있습니다. **podman login** 명령에 레지스트리 URL을 제공하지 않으면 구성된 기본 레지스트리에 인증됩니다.

```
[user@host ~]$ podman login registry.lab.example.com
Username: RH134
Password: EXAMPLEPASSWORD
Login Succeeded!
```

podman login 명령 **--username** 및 **--password-stdin** 옵션을 사용하여 레지스트리에 로그인 할 사용자 및 암호를 지정할 수도 있습니다. **--password-stdin** 옵션은 stdin에서 암호를 읽습니다. **--password** 옵션을 사용하여 암호를 직접 제공하지 않는 것이 좋습니다. 이 옵션은 암호를 로그 파일에 저장하기 때문입니다.

```
[user@host ~]# echo $PASSWORDVAR | podman login --username RH134 \
--password-stdin registry.access.redhat.com
```

레지스트리에 로그인했는지 확인하려면 **podman login** 명령 **--get-login** 옵션을 사용합니다.

```
[user01@rhel-vm ~]$ podman login registry.access.redhat.com --get-login
RH134
[user01@rhel-vm ~]$ podman login quay.io --get-login
Error: not logged into quay.io
```

위 출력에서 **podman** 유틸리티는 **RH134** 사용자 자격 증명을 사용하여 **registry.access.redhat.com** 레지스트리에 인증되지만 **podman** 유틸리티는 **quay.io** 레지스트리에 인증되지 않습니다.

컨테이너 레지스트리 구성

컨테이너 레지스트리의 기본 구성 파일은 **/etc/containers/registries.conf** 파일입니다.

```
[user@host ~]$ cat /etc/containers/registries.conf
# For more information on this configuration file, see containers-
registries.conf(5).
#
...output omitted...
```

```

unqualified-search-registries = ["registry.fedoraproject.org",
    "registry.access.redhat.com", "registry.centos.org", "quay.io", "docker.io"]

# [[registry]]
# # The "prefix" field is used to choose the relevant [[registry]] TOML table;
# # (only) the TOML table with the longest match for the input image name
# # (taking into account namespace/repo/tag/digest separators) is used.
#
# #
# # The prefix can also be of the form: *.example.com for wildcard subdomain
# # matching.
#
# #
# # If the prefix field is missing, it defaults to be the same as the "location"
# # field.
# prefix = "example.com/foo"
#
# #
# # If true, unencrypted HTTP as well as TLS connections with untrusted
# # certificates are allowed.
# insecure = false
#
# #
# # If true, pulling images with matching names is forbidden.
# blocked = false
#
# ...
...output omitted...

```

권한이 없는 사용자를 사용하여 컨테이너를 관리하는 것이 좋으므로 `$HOME/.config/containers` 디렉터리에서 컨테이너 레지스트리에 대한 `registries.conf` 파일을 생성할 수 있습니다. 이 디렉터리의 구성 파일은 `/etc/containers/registries.conf` 파일의 설정을 재정의하고 Podman이 rootless 모드로 실행될 때 사용됩니다.

`podman` 명령을 사용할 때 컨테이너 이미지의 정규화된 이름을 지정하지 않으면 컨테이너 이미지를 검색하는데 이 파일의 `unqualified-search-registries` 섹션에 있는 레지스트리 목록이 사용됩니다.

```
[user@host ~]$ podman pull ubi
```

명령줄에서 컨테이너 이미지의 정규화된 이름을 지정하면 컨테이너 유틸리티가 이 섹션에서 검색하지 않습니다. 컨테이너 이미지의 정규화된 이름을 사용하도록 `unqualified-search-registries` 섹션을 비워둘 수 있습니다.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/ubi:latest
```



참고

항상 컨테이너 이미지의 정규화된 이름을 사용하는 것이 좋습니다.

파일의 `[[registry]]` 섹션에서 컨테이너 레지스트리의 설정을 구성합니다. 각 컨테이너 레지스트리의 설정을 구성하려면 별도의 `[[registry]]` 섹션을 사용합니다.

```

[[registry]]
location = "registry.lab.example.com"
insecure = true
blocked = false

```

- **location** 설정은 컨테이너 레지스트리의 위치를 지정합니다.
- **insecure** 설정이 **true**로 설정된 경우 암호화되지 않은 HTTP 및 신뢰할 수 없는 인증서가 포함된 TLS 연결을 사용하여 레지스트리에 액세스할 수 있습니다.
- **blocked** 설정이 **true**로 설정된 경우 해당 레지스트리에서 이미지를 다운로드할 수 없습니다.

**참고**

이 강의실에서는 컨테이너 이미지를 제공하기 위해 Red Hat Quay를 기반으로 하는 프라이빗 비보안 레지스트리를 실행합니다. 이 레지스트리는 강의실 요구 사항을 충족합니다. 그러나 실제 시나리오에서는 비보안 레지스트리로 작업할 수 없습니다. 이 소프트웨어에 관한 자세한 내용은 <https://access.redhat.com/products/red-hat-quay>를 참조하십시오.

컨테이너 이미지 빌드에 필요한 컨테이너 파일

컨테이너 파일은 컨테이너 이미지 빌드에 필요한 명령이 포함된 텍스트 파일입니다. 컨테이너 파일에는 일반적으로 파일 및 디렉터리가 있는 경로 또는 URL을 정의하는 컨텍스트가 있습니다. 결과 컨테이너 이미지는 읽기 전용 계층으로 구성되며, 각 계층은 컨테이너 파일의 명령을 나타냅니다.

다음 출력은 **registry.access.redhat.com** 레지스트리의 UBI 이미지를 사용하고, **python3** 패키지를 설치하고, **hello** 문자열을 콘솔에 출력하는 컨테이너 파일의 예입니다.

```
[user@host ~]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python3
CMD ["/bin/bash", "-c", "echo hello"]
```

**참고**

컨테이너 파일의 생성 방법과 해당 사용 지침은 이 교육 과정의 범위를 벗어납니다. 컨테이너 파일에 대한 자세한 내용은 DO180 과정을 참조하십시오.

대규모 컨테이너 관리

최신 애플리케이션에서는 컨테이너를 사용하여 기능 구성 요소를 구현하는 경우가 점점 늘어나고 있습니다. 즉 컨테이너에서 서비스를 제공하고, 애플리케이션의 다른 부분에서는 이를 사용합니다. 조직에서 관리하는 컨테이너 수가 점점 많아지다 보면 작업량이 엄청나게 늘어날 수 있습니다.

프로덕션 환경에서 대규모로 컨테이너를 배포하려면 다음과 같은 문제에 맞게 환경을 조정할 수 있어야 합니다.

- 필수 서비스를 제공하는 컨테이너의 가용성을 플랫폼에서 보장해야 합니다.
- 해당 환경에서 실행 중인 컨테이너 수를 늘리거나 줄이고 트래픽을 부하 분산하여 애플리케이션 사용량 급증 현상에 대응할 수 있어야 합니다.
- 플랫폼에서 컨테이너 또는 호스트의 장애를 감지하고 그에 따라 대응해야 합니다.
- 개발자가 비교적 최신 애플리케이션 버전을 투명하고 안전하게 제공하려면 자동화된 워크플로가 필요할 수 있습니다.

Kubernetes는 컨테이너 호스트 클러스터에 컨테이너 기반 애플리케이션을 배포, 관리, 확장하는 오픈스택 레이션 서비스입니다. Kubernetes는 로드 밸런서를 사용하여 트래픽을 컨테이너로 리디렉션하므로 서비스

를 제공하는 컨테이너 수를 확장할 수 있습니다. 또한 Kubernetes는 사용자 정의 상태 점검을 지원하므로 컨테이너를 모니터링하고 실패할 경우 다시 시작할 수 있습니다.

Red Hat에서는 Red Hat OpenShift라는 Kubernetes 배포판을 제공합니다. Red Hat OpenShift는 Kubernetes 인프라를 토대로 빌드된 모듈식 구성 요소 및 서비스 집합이며, 원격 웹 기반 관리, 멀티테넌시, 모니터링 및 감사, 고급 보안 기능, 애플리케이션 라이프사이클 관리, 셀프 서비스 인스턴스 등 개발자를 위한 추가 기능을 제공합니다.

Red Hat OpenShift는 이 교육 과정의 범위를 벗어나지만, <https://www.openshift.com>에서 자세히 알아볼 수 있습니다.



참고

엔터프라이즈에서는 일반적으로 명령줄에서 개별 컨테이너를 실행하지 않습니다. 그 대신 Red Hat OpenShift와 같은 Kubernetes 기반 플랫폼을 사용하여 프로덕션에서 컨테이너를 실행하는 방법을 선호합니다.

그러나 컨테이너와 이미지를 수동으로 또는 소규모로 관리하려면 명령을 사용해야 할 수도 있습니다. 이 장에서는 컨테이너의 핵심 개념과 작동 방식, 유용성에 대한 이해를 높이기 위해 이러한 사용 사례에 중점을 둡니다.



참조

[cgroups\(7\)](#), [namespaces\(7\)](#), [seccomp\(2\)](#) 도움말 페이지

OCI(Open Container Initiative) 이미지 사양

<https://github.com/opencontainers/image-spec/blob/master/spec.md>

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index

에서 Starting with Containers 가이드의 Red Hat Enterprise Linux 9 Building, Running, and Managing Containers 장을 참조하십시오.

▶ 퀴즈

컨테이너 개념

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 컨테이너를 실행하는 Red Hat Enterprise Linux 툴은 무엇입니까?

- a. buildah
- b. container
- c. podman
- d. skopeo

▶ 2. 다음 중 컨테이너 기술을 설명하는 두 가지 항목은 무엇입니까? (두 개를 선택하십시오.)

- a. 컨테이너는 라이브러리 종속성을 추가하여 전체 운영 체제를 패키징합니다.
- b. 컨테이너는 시스템의 나머지 부분과 격리된 프로세스를 실행합니다.
- c. 각 컨테이너에는 자체 커널 버전과 라이브러리가 포함되어 있습니다.
- d. 컨테이너는 애플리케이션을 배포 및 관리하기 쉽도록 패키징하는 일반적인 방법입니다.

▶ 3. 다음 중 컨테이너 이미지에 관한 올바른 설명 두 가지는 무엇입니까? (두 개를 선택하십시오.)

- a. 컨테이너 이미지는 필요한 모든 런타임 종속성과 함께 애플리케이션을 패키징합니다.
- b. Docker에서 작동하는 컨테이너 이미지는 Podman에 사용할 수 없습니다.
- c. 컨테이너 이미지는 해당 이미지에 설치된 동일한 소프트웨어 버전이 있는 컨테이너 호스트에서만 실행할 수 있습니다.
- d. 컨테이너 이미지는 컨테이너를 만들기 위한 청사진 역할을 합니다.

▶ 4. Red Hat Enterprise Linux에서 컨테이너를 구현하는 데 사용되는 세 가지 핵심 기술은 무엇입니까? (세 개를 선택하십시오.)

- a. VM 호스팅을 위한 하이퍼바이저 코드
- b. 리소스 관리를 위한 cGroups(제어 그룹)
- c. 프로세스 분리를 위한 Namespaces
- d. 컨테이너 호스트와의 호환성을 위한 전체 운영 체제
- e. 보안을 위한 SELinux 및 Seccomp

▶ 5. 다음 중 컨테이너 파일에 관한 올바른 설명은 무엇입니까?

- a. 컨테이너 파일은 컨테이너를 실행하는 실행 파일입니다.
- b. 컨테이너 파일은 컨테이너 이미지를 빌드하는 실행 파일입니다.
- c. 컨테이너 파일은 컨테이너의 라이브러리 및 구성을 포함하는 압축 파일입니다.
- d. 컨테이너 파일은 컨테이너 빌드에 필요한 명령이 포함된 텍스트 파일입니다.
- e. 컨테이너 파일은 컨테이너 이미지 빌드에 필요한 명령이 포함된 텍스트 파일입니다.

▶ 솔루션

컨테이너 개념

다음 질문에 대한 올바른 답을 선택하십시오.

▶ 1. 다음 중 컨테이너를 실행하는 Red Hat Enterprise Linux 툴은 무엇입니까?

- a. buildah
- b. container
- c. podman
- d. skopeo

▶ 2. 다음 중 컨테이너 기술을 설명하는 두 가지 항목은 무엇입니까? (두 개를 선택하십시오.)

- a. 컨테이너는 라이브러리 종속성을 추가하여 전체 운영 체제를 패키징합니다.
- b. 컨테이너는 시스템의 나머지 부분과 격리된 프로세스를 실행합니다.
- c. 각 컨테이너에는 자체 커널 버전과 라이브러리가 포함되어 있습니다.
- d. 컨테이너는 애플리케이션을 배포 및 관리하기 쉽도록 패키징하는 일반적인 방법입니다.

▶ 3. 다음 중 컨테이너 이미지에 관한 올바른 설명 두 가지는 무엇입니까? (두 개를 선택하십시오.)

- a. 컨테이너 이미지는 필요한 모든 런타임 종속성과 함께 애플리케이션을 패키징합니다.
- b. Docker에서 작동하는 컨테이너 이미지는 Podman에 사용할 수 없습니다.
- c. 컨테이너 이미지는 해당 이미지에 설치된 동일한 소프트웨어 버전이 있는 컨테이너 호스트에서만 실행할 수 있습니다.
- d. 컨테이너 이미지는 컨테이너를 만들기 위한 청사진 역할을 합니다.

▶ 4. Red Hat Enterprise Linux에서 컨테이너를 구현하는 데 사용되는 세 가지 핵심 기술은 무엇인가? (세 개를 선택하십시오.)

- a. VM 호스팅을 위한 하이퍼바이저 코드
- b. 리소스 관리를 위한 cGroups(제어 그룹)
- c. 프로세스 분리를 위한 Namespaces
- d. 컨테이너 호스트와의 호환성을 위한 전체 운영 체제
- e. 보안을 위한 SELinux 및 Seccomp

▶ 5. 다음 중 컨테이너 파일에 관한 올바른 설명은 무엇입니까?

- a. 컨테이너 파일은 컨테이너를 실행하는 실행 파일입니다.
- b. 컨테이너 파일은 컨테이너 이미지를 빌드하는 실행 파일입니다.
- c. 컨테이너 파일은 컨테이너의 라이브러리 및 구성을 포함하는 압축 파일입니다.
- d. 컨테이너 파일은 컨테이너 빌드에 필요한 명령이 포함된 텍스트 파일입니다.
- e. 컨테이너 파일은 컨테이너 이미지 빌드에 필요한 명령이 포함된 텍스트 파일입니다.

컨테이너 배포

목표

레지스트리를 사용하여 이미지를 저장 및 검색하고, 컨테이너를 배포 및 쿼리하고 컨테이너에 액세스하는 컨테이너 관리 툴을 설명합니다.

Podman 유틸리티

Podman은 **container-tools** 메타 패키지의 모든 기능을 갖춘 컨테이너 엔진으로, OCI(Open Container Initiative) 컨테이너 및 이미지를 관리합니다. **podman** 유틸리티는 데몬을 사용하여 작동하지 않으므로 개발자는 시스템에 권한 있는 사용자 계정이 없어도 컨테이너를 시작하거나 중지할 수 있습니다. Podman은 컨테이너 및 이미지와 상호 작용하는 다양한 하위 명령을 제공합니다. 다음 목록은 이 섹션에서 사용되는 하위 명령을 보여줍니다.

Podman 명령

명령	설명
podman build	컨테이너 파일을 사용하여 컨테이너 이미지를 빌드합니다.
podman run	새 컨테이너에서 명령을 실행합니다.
podman images	로컬 스토리지의 이미지를 나열합니다.
podman ps	컨테이너에 대한 정보를 출력합니다.
podman inspect	컨테이너, 이미지, 볼륨, 네트워크 또는 포드의 구성 표시합니다.
podman pull	레지스트리에서 이미지를 다운로드합니다.
podman cp	컨테이너와 로컬 파일 시스템 간에 파일 또는 디렉터리를 복사합니다.
podman exec	실행 중인 컨테이너에서 명령을 실행합니다.
podman rm	하나 이상의 컨테이너를 제거합니다.
podman rmi	로컬에 저장된 하나 이상의 이미지를 제거합니다.
podman search	레지스트리에서 이미지를 검색합니다.

man 페이지를 사용하여 각 하위 명령에 대한 자세한 내용을 보려면 하이픈을 사용하여 **podman** 명령에 하위 명령을 추가하여 두 명령을 구분합니다. 예를 들어, **podman-build** man 페이지에서는 **podman build** 하위 명령의 사용에 대해 설명합니다.

이 강의의 주제를 다루기 위해 다음과 같은 시나리오를 상상해 보십시오.

시스템 관리자는 **python-38** 패키지를 사용하여 **python38**이라는 RHEL 8 UBI 컨테이너 이미지를 기반으로 하는 컨테이너를 실행해야 합니다. 또한 컨테이너 파일에서 컨테이너 이미지를 생성하고 해당 컨테이너 이미지에서 **python36**이라는 컨테이너를 실행해야 합니다. 컨테이너 파일로 생성한 컨테이너 이미지에

는 **python36:1.0** 태그가 있어야 합니다. 두 컨테이너의 차이점을 확인하십시오. 또한 컨테이너에 설치된 **python** 패키지가 로컬 시스템에 설치된 Python 버전과 충돌하지 않는지 확인하십시오.

컨테이너 유ти리티 설치

container-tools 메타 패키지에는 컨테이너 및 컨테이너 이미지와 상호 작용하는 데 필요한 유ти리티가 포함되어 있습니다. 시스템에서 컨테이너를 다운로드하고, 실행 및 비교하려면 **dnf install** 명령을 사용하여 **container-tools** 메타 패키지를 설치합니다. **dnf info** 명령을 사용하여 **container-tools** 패키지의 버전 및 콘텐츠를 확인합니다.

```
[root@host ~]# dnf install container-tools
...output omitted...
[user@host ~]$ dnf info container-tools
...output omitted...
Summary      : A meta-package which container tools such as podman, buildah,
               : skopeo, etc.
License       : MIT
Description   : Latest versions of podman, buildah, skopeo, runc, common, CRIU,
               : Uidica, etc as well as dependencies such as container-selinux
               : built and tested together, and updated.
...output omitted...
```

container-tools 메타 패키지는 할당된 작업을 수행하는 데 필요한 **podman** 및 **skopeo** 유ти리티를 제공합니다.

레지스트리에서 컨테이너 이미지 다운로드

먼저 **registry.redhat.io** 레지스트리에서 컨테이너를 검색하고 다운로드하도록 **podman** 유ти리티가 구성되어 있는지 확인합니다. **podman info** 명령은 구성된 레지스트리를 포함하여 **podman** 유ти리티에 관한 구성 정보를 표시합니다.

```
[user@host ~]$ podman info
...output omitted...
insecure registries:
  registries: []
registries:
  registries:
    - registry.redhat.io
    - quay.io
    - docker.io
...output omitted...
```

podman search 명령은 **registries.conf** 파일에 지정된 레지스트리 목록을 사용하여 일치하는 이름의 이미지를 검색합니다. 기본적으로 Podman은 정규화되지 않은 모든 검색 레지스트리에서 검색합니다.



참고

unqualified-search-registries 지시문은 **registry.redhat.io/ubi9/python-39** 와 같이 정규화되지 않은 이름의 이미지가 사용될 때 Podman이 이미지를 검색하거나 가져오는 데 사용하는 레지스트리 목록입니다. **containers-registries.conf(5)** man 페이지에서 자세한 내용을 확인할 수 있습니다.

레지스트리와 함께 구현되는 Docker 배포 API에 따라 일부 레지스트리는 검색 기능을 지원하지 않을 수 있습니다.

podman search 명령을 사용하여 구성된 레지스트리에서 **python-38** 패키지를 포함하는 이미지 목록을 표시합니다.

```
[user@host ~]$ podman search python-38
NAME                                     DESCRIPTION
registry.access.redhat.com/ubi7/python-38   Python 3.8 platform for building and
                                             running applications
registry.access.redhat.com/ubi8/python-38     Platform for building and running
                                             Python 3.8 applications
...output omitted...
```

`registry.access.redhat.com/ubi8/python-38` 이미지가 필수 컨테이너의 기준과 일치하는 것으로 표시됩니다.

skopeo inspect 명령을 사용하여 이미지를 다운로드하지 않고 로컬 디렉터리 또는 원격 레지스트리에서 다양한 컨테이너 이미지 형식을 검사할 수 있습니다. 이 명령 출력에는 사용 가능한 버전 태그 목록, 컨테이너화된 애플리케이션의 노출된 포트, 컨테이너 이미지의 메타데이터가 표시됩니다. **skopeo inspect** 명령을 사용하여 이미지에 필수 **python-38** 패키지가 포함되어 있는지 확인합니다.

```
[user@host ~]$ skopeo inspect docker://registry.access.redhat.com/ubi8/python-38
{
    "Name": "registry.access.redhat.com/ubi8/python-38",
    "Digest": "sha256:c6e522cba2cf2b3ae4a875d5210fb94aa1e7ba71b6cebd902a4f4df73cb090b8",
    "RepoTags": [
        ...output omitted...
        "1-68",
        "1-77-source",
        "latest"
    ],
    "...output omitted...
    "name": "ubi8/python-38",
    "release": "86.1648121386",
    "summary": "Platform for building and running Python 3.8 applications",
    "...output omitted...
}
```

`registry.access.redhat.com/ubi8/python-38` 이미지에 필수 패키지가 포함되어 있으며 필수 이미지를 기반으로 합니다. **podman pull** 명령을 사용하여 선택한 이미지를 로컬 시스템에 다운로드합니다. 위 출력에 있는 이미지의 정규화된 이름을 사용하여 컨테이너 버전 또는 레지스트리에 대한 모호성을 방지할 수 있습니다.

```
[user@host ~]$ podman pull registry.access.redhat.com/ubi8/python-38
Trying to pull registry.access.redhat.com/ubi8/python-38:latest...
Getting image source signatures
Checking if image destination supports signatures
Copying blob c530010fb61c done
...output omitted...
```

그런 다음 **podman images** 명령을 사용하여 로컬 이미지를 표시합니다.

```
[user@host ~]$ podman images
REPOSITORY                                     TAG      IMAGE ID      CREATED       SIZE
registry.access.redhat.com/ubi8/python-38    latest   a33d92f90990  1 hour ago  901 MB
```

컨테이너 파일에서 컨테이너 이미지 생성

`python36-app` 디렉터리에 컨테이너 이미지를 생성할 수 있도록 다음 컨테이너 파일이 제공됩니다.

```
[user@host python36-app]$ cat Containerfile
FROM registry.access.redhat.com/ubi8/ubi:latest
RUN dnf install -y python36
CMD ["/bin/bash", "-c", "sleep infinity"]
```

위 컨테이너 파일에서는 `registry.access.redhat.com/ubi8/ubi:latest` 이미지를 기본 이미지로 사용합니다. 그런 다음 컨테이너 파일은 `python36` 패키지를 설치하고 `sleep infinity` Bash 명령을 실행하여 컨테이너가 종료되지 않도록합니다.

일반적으로 컨테이너는 프로세스를 실행한 다음 해당 프로세스가 완료되면 종료됩니다. 프로세스가 완료되지 않으므로 `sleep infinity` 명령은 컨테이너가 종료되지 않도록합니다. 그러면 컨테이너 내부에서 테스트, 개발, 디버그를 수행할 수 있습니다.

컨테이너 파일을 검사한 후 `podman build` 명령을 사용하여 이미지를 빌드합니다. `podman build` 명령의 구문은 다음과 같습니다.

```
[user@host ~]$ podman build -t NAME:TAG DIR
```

NAME

새 이미지의 이름입니다.

TAG

새 이미지에 대한 태그입니다. 태그를 지정하지 않으면 이미지에 `latest` 태그가 자동으로 지정됩니다.

DIR

작업 디렉터리의 경로입니다. 컨테이너 파일은 작업 디렉터리에 있어야 합니다. 작업 디렉터리가 현재 디렉터리인 경우 점(.)으로 지정합니다. 현재 디렉터리와 다른 디렉터리를 지정하려면 `-f` 플래그를 사용합니다.

다음 예에서는 `podman build` 명령 `-t` 옵션을 사용하여 새 이미지에 이름 `python36` 과 태그 `1.0` 을 제공할 수 있습니다. 컨테이너 파일은 현재 디렉터리에 있습니다.

```
[user@host python36-app]$ podman build -t python36:1.0 .
STEP 1/3: FROM registry.access.redhat.com/ubi8/ubi:latest
STEP 2/3: RUN dnf install -y python36
...output omitted...
STEP 3/3: CMD ["/bin/bash", "-c", "sleep infinity"]
COMMIT python36:1.0
--> 35ab820880f
Successfully tagged localhost/python36:1.0
35ab820880f1708fa310f835407ffc94cb4b4fe2506b882c162a421827b156fc
```

위 출력의 마지막 행에는 컨테이너 이미지 ID가 표시됩니다. 대부분의 Podman 명령은 컨테이너 이미지 ID의 처음 12자를 사용하여 컨테이너 이미지를 참조합니다. 이 짧은 ID나 컨테이너 또는 컨테이너 이미지의 이름을 대부분의 Podman 명령에 인수로 사용할 수 있습니다.

**참고**

태그에 버전 번호가 지정되지 않은 경우 :latest 태그를 사용하여 이미지가 생성됩니다. 이미지 이름이 지정되지 않은 경우 이미지 및 태그 필드에 <none> 문자열이 표시됩니다.

`podman images` 명령을 사용하여 정의된 이름 및 태그로 이미지가 생성되었는지 확인합니다.

```
[user@host ~]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED       SIZE
localhost/python36   1.0      35ab820880f1  3 minute ago  266 MB
registry.access.redhat.com/ubi8/python-38 latest  a33d92f90990  1 hour ago   901 MB
```

그런 다음 `podman inspect` 명령을 사용하여 컨테이너 이미지의 하위 수준 정보를 보고 해당 콘텐츠가 컨테이너 요구 사항과 일치하는지 확인합니다.

```
[user@host ~]$ podman inspect localhost/python36:1.0
...
{
    "Cmd": [
        "/bin/bash",
        "-c",
        "sleep infinity"
    ],
    ...
    {
        "created": "2022-04-18T19:47:52.708227513Z",
        "created_by": "/bin/sh -c dnf install -y python36",
        "comment": "FROM registry.access.redhat.com/ubi8/ubi:latest"
    },
    ...
}
```

`podman inspect` 명령의 출력에는 `registry.access.redhat.com/ubi8/ubi:latest` 기본 이미지, `python36` 패키지를 설치하는 `dnf` 명령, 컨테이너가 종료되지 않도록 런타임에 실행되는 `sleep infinity` Bash 명령이 표시됩니다.

**참고**

`podman inspect` 명령은 `python-38` 이미지에서 `python36` 이미지까지 다양하게 출력합니다. `/python36` 이미지는 `registry.access.redhat.com/ubi8/ubi:latest` 기본 이미지에 변경 사항이 있는 계층을 추가하여 생성한 반면, `python-38` 이미지는 그 자체가 기본 이미지이기 때문입니다.

컨테이너 실행

이제 필수 컨테이너 이미지가 있으므로 해당 이미지를 사용하여 컨테이너를 실행할 수 있습니다. 컨테이너의 상태는 다음 중 하나일 수 있습니다.

생성됨

생성되었지만 시작되지 않은 컨테이너입니다.

실행

프로세스와 함께 실행되고 있는 컨테이너입니다.

중지됨

프로세스가 중지된 컨테이너입니다.

일시 중지됨

프로세스가 일시 중지된 컨테이너입니다. rootless 컨테이너에는 지원되지 않습니다.

삭제됨

프로세스가 만료된 컨테이너입니다.

podman ps 명령은 시스템에서 실행 중인 컨테이너를 나열합니다. 시스템의 모든 컨테이너(생성됨, 중지됨, 일시 중지됨 또는 실행 중 상태)를 보려면 **podman ps -a** 명령을 사용하십시오.

podman create 명령은 나중에 실행할 컨테이너를 생성하는 데 사용합니다. 컨테이너를 생성하려면 **localhost/python36** 컨테이너 이미지의 ID를 사용합니다. 또한 **--name** 옵션을 사용하여 컨테이너를 식별하는 이름을 설정합니다. 이 명령은 컨테이너의 긴 ID를 출력합니다.

```
[user@host ~]$ podman create --name python36 dd6ca291f097
c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec
```

**참고**

podman create 또는 **podman run** 명령을 **--name** 옵션과 함께 사용하여 컨테이너의 이름을 설정하지 않으면 **podman** 유틸리티에서 컨테이너에 임의의 이름을 할당합니다.

그런 다음 **podman ps** 및 **podman ps -a** 명령을 사용하여 컨테이너가 생성되었지만 시작되지 않았는지 확인합니다. 짧은 ID, 이름, 컨테이너 상태, 시작 시 컨테이너가 실행하는 명령, 컨테이너를 생성하는 이미지 등 **python36** 컨테이너에 대한 정보를 확인할 수 있습니다.

```
[user@host ~]$ podman create --name python36 dd6ca291f097
c54c7ee281581c198cb96b07d78a0f94be083ae94dacbae69c05bd8cd354bbec
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
[user@host ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
c54c7ee28158 localhost/python36:1.0 /bin/bash -c slee... 5 seconds ago Created
python36
```

컨테이너가 올바르게 생성되었는지 확인한 후에는 컨테이너를 시작하기로 결정하고 **podman start** 명령을 실행합니다. 이름 또는 컨테이너 ID를 사용하여 컨테이너를 시작할 수 있습니다. 이 명령은 컨테이너의 이름을 출력합니다.

```
[user@host ~]$ podman start python36
python36
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND CREATED STATUS
PORTS NAMES
c54c7ee28158 localhost/python36:1.0 /bin/bash -c slee... 6 minutes ago Up 3
seconds ago python36
```

원격 리포지토리에서 컨테이너 실행

`podman run` 명령을 사용하면 한 번에 컨테이너를 생성하고 실행할 수 있습니다. `podman run` 명령은 컨테이너 내부에서 프로세스를 실행하고 이 프로세스는 새 컨테이너를 시작합니다.

`podman run` 명령 `-d` 옵션을 사용하면 컨테이너가 세션의 포그라운드가 아닌 백그라운드에서 실행되는 분리 모드에서 실행됩니다. `python36` 컨테이너 예제에서는 실행할 컨테이너에 대한 명령을 제공할 필요가 없습니다. 해당 컨테이너에 대한 이미지를 생성한 컨테이너 파일에 `sleep infinity` 명령이 이미 제공되어 있기 때문입니다.

`python38` 컨테이너를 생성하기 위해 `podman run` 명령을 사용하고 `registry.access.redhat.com/ubi8/python-38` 이미지를 참조하기로 결정합니다. 또한 컨테이너가 종료되지 않도록 `sleep infinity` 명령을 사용하기로 결정합니다.

```
[user@host ~]$ podman run -d --name python38 \
registry.access.redhat.com/ubi8/python-38 \
sleep infinity

a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
c54c7ee28158 localhost/python36:1.0 /bin/bash -c
slee... 37 minutes ago Up 30 minutes ago python36
a60f71a1dc1b registry.access.redhat.com/ubi8/python-38:latest sleep infinity
32 seconds ago Up 33 seconds ago python38
```



중요

정규화된 이미지 이름을 사용하여 컨테이너를 실행했는데 해당 이미지가 아직 로컬에 저장되지 않은 경우, `podman run` 명령으로 먼저 레지스트리에서 이미지를 가져온 다음 실행해야 합니다.

컨테이너의 환경 격리

컨테이너는 애플리케이션의 환경을 격리합니다. 각 컨테이너에는 자체 파일 시스템, 네트워킹, 프로세스가 있습니다. `ps` 명령의 출력을 확인하고 호스트 시스템과 실행 중인 컨테이너를 비교하면 격리 기능을 확인할 수 있습니다.

먼저 로컬 시스템에서 `ps -ax` 명령을 실행하면 이 명령은 예상 결과와 다양한 프로세스를 반환합니다.

```
[root@host ~]# ps -ax
 PID TTY      STAT   TIME COMMAND
  1 ?        Ss     0:01 /usr/lib/systemd/systemd --switched-root --system --
deseriali
  2 ?        S      0:00 [kthreadd]
  3 ?        I<    0:00 [rcu_gp]
  4 ?        I<    0:00 [rcu_par_gp]
...output omitted...
```

`podman exec` 명령은 실행 중인 컨테이너 내에서 명령을 실행합니다. 명령은 컨테이너의 이름 또는 ID를 첫 번째 인수로 사용하고 컨테이너 내부에서 실행할 명령을 다음 인수로 사용합니다. `podman exec` 명령을

사용하여 **python36** 컨테이너에서 실행 중인 프로세스를 확인합니다. **ps -ax** 명령의 출력은 로컬 시스템과 다른 프로세스를 실행하고 있기 때문에 다르게 표시됩니다.

```
[student@host ~]$ podman exec python38 ps -ax
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss      0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /
/usr/bin/sleep infinity
    7 ?        R      0:00 ps -ax
```

sh -c 명령을 사용하여 컨테이너에서 실행할 명령을 캡슐화할 수 있습니다. 다음 예제에서 **ps -ax > /tmp/process-data.log** 명령은 컨테이너에서 실행할 명령으로 해석됩니다. 명령을 캡슐화하지 않으면 Podman에서 보다 큼 문자(>)를 **podman exec** 옵션에 대한 인수 대신 **podman** 명령의 일부로 해석할 수 있습니다.

```
[student@host ~]$ podman exec python38 sh -c 'ps -ax > /tmp/process-data.log'
  PID TTY      STAT   TIME COMMAND
    1 ?        Ss      0:00 /usr/bin/coreutils --coreutils-prog-shebang=sleep /
/usr/bin/sleep infinity
    7 ?        R      0:00 ps -ax
```

호스트 시스템에 설치된 **python** 버전을 컨테이너에 설치된 **python** 버전과 비교하기로 결정합니다.

```
[user@host ~]$ python3 --version
Python 3.9.10
[user@host ~]$ podman exec python36 python3 --version
Python 3.6.8
[user@host ~]$ podman exec python38 python3 --version
Python 3.8.8
```

컨테이너의 파일 시스템 경리

개발자는 파일 시스템 경리 기능을 사용하여 여러 물리 또는 가상 시스템을 사용하지 않고도 다양한 버전의 프로그래밍 언어에 대한 애플리케이션을 작성하고 테스트할 수 있습니다.

/tmp 디렉터리의 터미널에 **hello world**를 표시하는 간단한 Bash 스크립트를 생성합니다.

```
[user@host ~]$ echo "echo 'hello world'" > /tmp/hello.sh
```

/tmp/hello.sh 파일은 호스트 시스템에 있으며 컨테이너 내부의 파일 시스템에는 없습니다. **podman exec**를 사용하여 스크립트를 실행하려고 하면 **/tmp/hello.sh** 스크립트가 컨테이너에 없기 때문에 오류가 발생합니다.

```
[user@host ~]$ stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19          Blocks: 8          IO Block: 4096   regular file
Device: fc04h/64516d Inode: 17655599      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/ user)  Gid: ( 1000/ user)
Context: unconfined_u:object_r:user_t:object_t
Access: 2022-04-19 21:47:40.101601412 -0400
Modify: 2022-04-19 21:47:36.497558132 -0400
Change: 2022-04-19 21:47:36.497558132 -0400
 Birth: 2022-04-19 21:45:24.785976758 -0400
```

```
[user@host ~]$ podman exec python38 stat /tmp/hello.sh
stat: cannot statx '/tmp/hello.sh': No such file or directory
```

podman cp 명령은 호스트와 컨테이너 파일 시스템 간에 파일 및 디렉터리를 복사합니다. **podman cp** 명령을 사용하여 `/tmp/hello.sh` 파일을 `python38` 컨테이너에 복사할 수 있습니다.

```
[user@host ~]$ podman cp /tmp/hello.sh python38:/tmp/hello.sh

[user@host ~]$ podman exec python38 stat /tmp/hello.sh
  File: /tmp/hello.sh
  Size: 19          Blocks: 8          IO Block: 4096   regular file
Device: 3bh/59d Inode: 12280058      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1001/ default)  Gid: (     0/    root)
Access: 2022-04-20 01:47:36.000000000 +0000
Modify: 2022-04-20 01:47:36.000000000 +0000
Change: 2022-04-20 02:02:04.732982187 +0000
 Birth: 2022-04-20 02:02:04.732982187 +0000
```

스크립트를 컨테이너 파일 시스템에 복사하면 컨테이너 내에서 실행할 수 있습니다.

```
[user@host ~]$ podman exec python38 bash /tmp/hello.sh
hello world
```

컨테이너 및 이미지 제거

podman rm 및 **podman rmi** 명령을 각각 사용하여 컨테이너와 이미지를 제거할 수 있습니다. 컨테이너 이미지를 제거하기 전에 해당 이미지에서 기존에 실행 중인 컨테이너를 제거해야 합니다.

`python38` 컨테이너 및 관련 이미지를 제거하기로 결정합니다. `python38` 컨테이너가 있는 `registry.access.redhat.com/ubi8/python-38` 이미지를 제거하려고 하면 오류가 발생합니다.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Error: Image used by
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462: image is in use
by a container
```

컨테이너를 중지해야 제거할 수 있습니다. 컨테이너를 중지하려면 **podman stop** 명령을 사용합니다.

```
[user@host ~]$ podman stop python38
```

컨테이너를 중지한 후 **podman rm** 명령을 사용하여 컨테이너를 제거합니다.

```
[user@host ~]$ podman rm python38
a60f71a1dc1b997f5ef244aaed232e5de71dd1e8a2565428ccfebde73a2f9462
```

컨테이너가 더 이상 존재하지 않으면 **podman rmi** 명령을 사용하여 `registry.access.redhat.com/ubi8/python-38` 이미지를 제거할 수 있습니다.

```
[user@host ~]$ podman rmi registry.access.redhat.com/ubi8/python-38
Untagged: registry.access.redhat.com/ubi8/python-38:latest
Deleted: a33d92f90990c9b1bad9aa98fe017e48f30c711b49527dcc797135352ea57d12
```



참조

podman(1), podman-build(1), podman-cp(1), podman-exec(1), podman-images(1), podman-inspect(1), podman-ps(1), podman-pull(1), podman-rm(1), podman-rmi(1), podman-run(1), podman-search(1), podman-stop(1) 도움말 폐이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index#starting-with-containers_building-running-and-managing-containers

에서 Starting with Containers 가이드의 Building, Running, and Managing Linux Containers on Red Hat Enterprise Linux 9 장을 참조하십시오.

▶ 연습 가이드

컨테이너 배포

이 연습에서는 컨테이너 관리 툴을 사용하여 이미지를 빌드하고, 컨테이너를 실행하고, 실행 중인 컨테이너 환경을 쿼리합니다.

결과

- 컨테이너 이미지 레지스트리를 구성하고 기존 이미지에서 컨테이너를 생성합니다.
- 컨테이너 파일을 사용하여 컨테이너 생성
- 호스트 시스템에서 컨테이너로 스크립트를 복사하여 해당 스크립트를 실행합니다.
- 컨테이너 및 이미지를 삭제합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start containers-deploy
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **container-tools** 메타 패키지를 설치합니다.

```
[student@servera ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- ▶ 3. 홈 디렉터리에 **registry.lab.example.com** 강의실 레지스트리를 구성합니다.
redhat321을 암호로 사용하여 **admin** 사용자로 컨테이너 레지스트리에 로그인합니다.

- 3.1. **/home/student/.config/containers** 디렉터리를 생성합니다.

```
[student@servera ~]$ mkdir -p /home/student/.config/containers
```

- 3.2. 다음 콘텐츠를 사용하여 `/home/student/.config/containers/registries.conf`라는 새 파일을 만듭니다.

```
unqualified-search-registries = ['registry.lab.example.com']

[[registry]]
location = "registry.lab.example.com"
insecure = true
blocked = false
```

- 3.3. 강의실 레지스트리가 추가되었는지 확인합니다.

```
[student@servera ~]$ podman info
...output omitted...
registries:
  registry.lab.example.com:
    Blocked: false
    Insecure: true
    Location: registry.lab.example.com
    MirrorByDigestOnly: false
    Mirrors: null
    Prefix: registry.lab.example.com
search:
  - registry.lab.example.com
...output omitted...
```

- 3.4. 강의실 레지스트리에 로그인합니다.

```
[student@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- ▶ 4. `python 3.8` 패키지가 있고 `ubi8` 이미지를 기반으로 하는 이미지에서 `python38` 컨테이너를 분리 모드로 실행합니다. 이미지는 원격 레지스트리에서 호스팅됩니다.

- 4.1. `registry.lab.example.com` 레지스트리에서 `python-38` 컨테이너를 검색합니다.

```
[student@servera ~]$ podman search registry.lab.example.com/
NAME                                     DESCRIPTION
...output omitted...
registry.lab.example.com/ubi8/python-38
registry.lab.example.com/ubi8/httpd-24
registry.lab.example.com/rhel8/php-74
```

- 4.2. 이미지를 검사합니다.

```
[student@servera ~]$ skopeo inspect \
docker://registry.lab.example.com/ubi8/python-38
...output omitted...
  "description": "Python 3.8 available as container is a base platform for
building and running various Python 3.8 applications and frameworks.
...output omitted...
```

4.3. `python-38` 컨테이너 이미지를 가져옵니다.

```
[student@servera ~]$ podman pull registry.lab.example.com/ubi8/python-38
Trying to pull registry.lab.example.com/ubi8/python-38:latest...
...output omitted...
671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

4.4. 컨테이너가 로컬 이미지 리포지토리에 다운로드되었는지 확인합니다.

<code>podman images</code>					
REPOSITORY	TAG	IMAGE ID	CREATED	SIZE	
registry.lab.example.com/ubi8/python-38	latest	671cc3cb4298	5 days ago	901 MB	

4.5. `python38` 컨테이너를 시작합니다.

```
[student@servera ~]$ podman run -d --name python38 \
registry.lab.example.com/ubi8/python-38 sleep infinity
004756b52d3d3326545f5075594cffa858afd474b903288723a3aa299e72b1af
```

4.6. 해당 컨테이너가 삭제되었는지 확인합니다.

<code>podman ps</code>					
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS NAMES
004756b52d3d	registry.lab.example.com/ubi8/python-38:latest	sleep infinity	About a minute ago	Up About a minute ago	python38

- ▶ 5. 컨테이너 파일에서 `python39:1.0`이라는 컨테이너 이미지를 빌드하고 이 이미지를 사용하여 `python39`이라는 컨테이너를 생성합니다.

5.1. `/home/student/python39` 디렉터리에서 컨테이너 파일을 검사합니다.

```
[student@servera ~]$ cat /home/student/python39/Containerfile
FROM registry.lab.example.com/ubi9-beta/ubi:latest
RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms]\nbaseurl = http://
content.example.com/rhel9.0/x86_64/dvd/BaseOS\nenabled = true\npgpcheck =
false\nname = Red Hat Enterprise Linux 9.0 BaseOS (dvd)\n[rhel-9.0-for-x86_64-
appstream-rpms]\nbaseurl = http://content.example.com/rhel9.0/x86_64/dvd/AppStream
\nenabled = true\npgpcheck = false\nname = Red Hat Enterprise Linux 9.0 Appstream
(dvd)'/>/etc/yum.repos.d/rhel_dvd.repo
RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_64-baseos-rpms --
enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3
```

5.2. 컨테이너 파일에서 컨테이너 이미지를 생성합니다.

```
[student@servera ~]$ podman build -t python39:1.0 /home/student/python39/.
STEP 1/4: FROM registry.lab.example.com/ubi9-beta/ubi:latest
...output omitted...
STEP 2/4: RUN echo -e '[rhel-9.0-for-x86_64-baseos-rpms] ...
...output omitted...
STEP 3/4: RUN yum install --disablerepo=* --enablerepo=rhel-9.0-for-x86_64-baseos-
rpms --enablerepo=rhel-9.0-for-x86_64-appstream-rpms -y python3
...output omitted...
STEP 4/4: CMD ["/bin/bash", "-c", "sleep infinity"]
...output omitted...
Successfully tagged localhost/python39:1.0
80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
```

5.3. 해당 컨테이너 이미지가 로컬 이미지 리포지토리에 있는지 확인합니다.

```
[student@servera ~]$ podman images
REPOSITORY                      TAG      IMAGE ID      CREATED        SIZE
localhost/python39                1.0      80e68c195925  3 minutes ago  266 MB
registry.lab.example.com/ubi8/python-38  latest   671cc3cb4298  5 days ago    901 MB
registry.lab.example.com/ubi9-beta/ubi    latest   fca12da1dc30  4 months ago   235 MB
```

5.4. `python39` 컨테이너를 검사합니다.

```
[student@servera ~]$ podman inspect localhost/python39:1.0
...output omitted...
  "comment": "FROM registry.lab.example.com/ubi9-beta/ubi:latest"
...output omitted...
  "created_by": "/bin/sh -c yum install --disablerepo=*
--enablerepo=rhel-9.0-for-x86_64-baseos-rpms --enablerepo=rhel-9.0-for-x86_64-
appstream-rpms -y python3"
...output omitted...
  "created_by": "/bin/sh -c #(nop) CMD [\"/bin/bash\", \"-c\", \"sleep
infinity\"]",
...output omitted...
```

5.5. `python39` 컨테이너를 생성합니다.

```
[student@servera ~]$ podman create --name python39 localhost/python39:1.0
3db4eabe9043224a7bdf195ab5fd810bf95db98dc29193392cef7b94489e1aae
```

5.6. `python39` 컨테이너를 시작합니다.

```
[student@servera ~]$ podman start python39
python39
```

5.7. 컨테이너가 실행되고 있는지 확인합니다.

```
[student@servera ~]$ podman ps
CONTAINER ID  IMAGE                               COMMAND
CREATED      STATUS      PORTS     NAMES
004756b52d3d  registry.lab.example.com/ubi8/python-38:latest  sleep infinity
              33 minutes ago   Up 33 minutes ago
3db4eabe9043  localhost/python39:1.0           /bin/bash -c
              slee... About a minute ago  Up 42 seconds ago
                                         python39
```

- ▶ 6. 실행 중인 컨테이너의 /tmp 디렉터리에 /home/student/script.py 스크립트를 복사하고 이 스크립트를 각 컨테이너에서 실행합니다.

- 6.1. /home/student/script.py python 스크립트를 두 컨테이너의 /tmp 디렉터리에 복사합니다.

```
[student@servera ~]$ podman cp /home/student/script.py python39:/tmp/script.py
[student@servera ~]$ podman cp /home/student/script.py python38:/tmp/script.py
```

- 6.2. 두 컨테이너 모두에서 Python 스크립트를 실행한 다음 호스트에서 Python 스크립트를 실행합니다.

```
[student@servera ~]$ podman exec -it python39 python3 /tmp/script.py
This script was not run on the correct version of Python
Expected version of Python is 3.8
Current version of python is 3.9
[student@servera ~]$ podman exec -it python38 python3 /tmp/script.py
This script was correctly run on Python 3.8
[student@servera ~]$ python3 /home/student/script.py
This script was not run on the correct version of Python
Expected version of Python is 3.8
Current version of python is 3.9
```

- ▶ 7. 컨테이너 및 이미지를 삭제합니다. workstation으로 돌아갑니다.

- 7.1. 두 컨테이너를 모두 중지합니다.

```
[student@servera ~]$ podman stop python39 python38
...output omitted...
python38
python39
```

- 7.2. 두 컨테이너를 모두 제거합니다.

```
[student@servera ~]$ podman rm python39 python38
3db4eabe9043224a7bdf195ab5fd810bf95db98dc29193392cef7b94489e1aae
004756b52d3d3326545f5075594cffa858af474b903288723a3aa299e72b1af
```

- 7.3. 두 컨테이너 이미지를 모두 제거합니다.

```
[student@servera ~]$ podman rmi localhost/python39:1.0 \
registry.lab.example.com/ubi8/python-38:latest \
registry.lab.example.com/ubi9-beta/ubi
Untagged: localhost/python39:1.0
Untagged: registry.lab.example.com/ubi8/python-38:latest
Deleted: 80e68c195925beafe3b2ad7a54fe1e5673993db847276bc62d5f9d109e9eb499
Deleted: 219e43f6ff96fd11ea64f67cd6411c354dacbc5cbe296ff1fdbf5b717f01d89a
Deleted: 671cc3cb42984e338733ebb5a9a68e69e267cb7f9cb802283d3bc066f6321617
```

7.4. `workstation`으로 돌아갑니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish containers-deploy
```

이것으로 섹션을 완료합니다.

컨테이너 스토리지 및 네트워크 리소스 관리

목표

컨테이너 호스트에서 스토리지를 공유하여 컨테이너 데이터를 위한 영구저장장치를 제공하고 컨테이너 네트워크를 구성합니다.

컨테이너 리소스 관리

간단한 프로세스를 실행하고 종료하는 데 컨테이너를 사용할 수 있습니다.

데이터베이스 서버와 같은 서비스를 지속적으로 실행하도록 컨테이너를 구성할 수도 있습니다. 서비스를 지속적으로 실행하는 경우 결국 영구저장장치 또는 추가 네트워크에 대한 액세스 등 더 많은 리소스를 컨테이너에 추가해야 할 수 있습니다.

다양한 전략을 사용하여 컨테이너의 영구 스토리지를 구성할 수 있습니다.

- Red Hat OpenShift와 같은 엔터프라이즈 컨테이너 플랫폼에 대한 대규모 배포의 경우 정교한 스토리지 솔루션을 사용하여 기본 인프라를 몰라도 컨테이너에 스토리지를 제공할 수 있습니다.
- 확장할 필요가 없는 단일 컨테이너 호스트에 대한 소규모 배포의 경우 실행 중인 컨테이너에 마운트할 디렉터리를 생성하여 컨테이너 호스트에서 영구 스토리지를 생성할 수 있습니다.

웹 서버 또는 데이터베이스 서버와 같은 컨테이너에서 컨테이너 호스트 외부의 클라이언트에 콘텐츠를 제공하는 경우, 해당 클라이언트가 컨테이너 콘텐츠에 액세스할 수 있도록 통신 채널을 설정해야 합니다. 컨테이너에 대한 통신을 활성화하도록 포트 매핑을 구성할 수 있습니다. 포트 매핑을 사용하면 컨테이너 호스트의 포트를 대상으로 하는 요청이 컨테이너 내부의 포트로 전달됩니다.

다음 작업을 수행해야 한다고 가정합니다.

- MariaDB를 기반으로 **db01**이라는 컨테이너화된 데이터베이스를 생성합니다.
- 포트 3306/tcp에서 트래픽을 허용하도록 컨테이너 포트 매핑 및 호스트 방화벽을 구성합니다.
- 적절한 SELinux 컨텍스트에서 영구 스토리지를 사용하도록 **db01** 컨테이너를 구성합니다.
- client01** 컨테이너가 DNS를 사용하여 **db01** 컨테이너와 통신할 수 있도록 적절한 네트워크 구성을 추가합니다.

컨테이너의 환경 변수

일부 컨테이너 이미지에서는 생성 시 환경 변수를 전달하여 컨테이너를 사용자 지정할 수 있습니다. 환경 변수를 사용하면 고유한 사용자 지정 이미지를 생성하지 않고도 컨테이너에 매개 변수를 설정하여 환경에 맞게 조정할 수 있습니다. 일반적으로 컨테이너 이미지는 이미지에 계층을 추가하여 유지 관리하기 어려울 수 있으므로 수정하지 않습니다.

`podman run -d registry.lab.example.com/rhel8/mariadb-105` 명령을 사용하여 컨테이너화된 데이터베이스를 실행했지만 컨테이너가 시작되지 않았습니다.

```
[user@host ~]$ podman run -d registry.lab.example.com/rhel8/mariadb-105 \
--name db01
20751a03897f14764fb0e7c58c74564258595026124179de4456d26c49c435ad
[user@host ~]$ podman ps -a
CONTAINER ID   IMAGE               COMMAND
CREATED        STATUS              PORTS
NAMES
20751a03897f   registry.lab.example.com/rhel8/mariadb-105:latest   run-mysqld
29 seconds ago  Exited (1) 29 seconds ago          db01
```

`podman container logs` 명령을 사용하여 컨테이너 상태의 원인을 조사합니다.

```
[user@host ~]$ podman container logs db01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[_a-zA-Z0-9]+$')
  MYSQL_PASSWORD (regex: '^[_a-zA-Z0-9~-!@#$%^&*()-=<>,_.?;:_]+$')
  MYSQL_DATABASE (regex: '^[_a-zA-Z0-9]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[_a-zA-Z0-9~-!@#$%^&*()-=<>,_.?;:_]+$')
Or both.
...output omitted...
```

위 출력에서 필수 환경 변수가 컨테이너에 전달되지 않았기 때문에 컨테이너가 계속 실행되지 않았음을 확인합니다. 따라서 `mariadb-105` 컨테이너 이미지를 검사하여 컨테이너를 사용자 지정하는 환경 변수에 대한 자세한 정보를 확인합니다.

```
[user@host ~]$ skopeo inspect docker://registry.lab.example.com/rhel8/mariadb-105
...output omitted...
{
  "name": "rhel8/mariadb-105",
  "release": "40.1647451927",
  "summary": "MariaDB 10.5 SQL database server",
  "url": "https://access.redhat.com/containers/#/registry.access.redhat.com/rhel8/mariadb-105/images/1-40.1647451927",
  "usage": "podman run -d -e MYSQL_USER=user -e MYSQL_PASSWORD=pass -e MYSQL_DATABASE=db -p 3306:3306 rhel8/mariadb-105",
  "vcs-ref": "c04193b96a119e176ada62d779bd44a0e0edf7a6",
  "vcs-type": "git",
  "vendor": "Red Hat, Inc.",
  ...output omitted...
```

출력에 있는 `usage` 레이블은 이미지 실행 방법의 예를 제공합니다. `url` 레이블은 환경 변수 및 컨테이너 이미지 사용 방법에 관한 기타 정보를 설명하는 Red Hat Container Catalog의 웹 페이지를 가리킵니다.

이 이미지에 대한 설명서는 컨테이너에서 데이터베이스 서비스에 3306 포트를 사용함을 보여줍니다. 설명서는 또한 다음 환경 변수를 사용하여 데이터베이스 서비스를 구성할 수도 있음을 보여줍니다.

mariadb 이미지의 환경 변수

변수	설명
MYSQL_USER	생성할 MySQL 계정의 사용자 이름

변수	설명
MYSQL_PASSWORD	사용자 계정의 암호
MYSQL_DATABASE	데이터베이스 이름
MYSQL_ROOT_PASSWORD	root 사용자의 암호(옵션)

해당 이미지에 사용 가능한 환경 변수를 검사한 후 **podman run** 명령 **-e** 옵션을 사용하여 환경 변수를 컨테이너에 전달하고 **podman ps** 명령을 사용하여 컨테이너가 실행되는지 확인합니다.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
4b8f01be7fd6 registry.lab.example.com/rhel8/mariadb-105:latest run-mysqld 6
seconds ago Up 6 seconds ago db01
```

컨테이너 영구 스토리지

기본적으로 컨테이너를 실행하면 모든 콘텐츠에서 컨테이너 기반 이미지를 사용합니다. 컨테이너 이미지의 임시 특성을 고려할 때 컨테이너를 제거하면 사용자 또는 애플리케이션이 작성하는 새 데이터가 모두 손실됩니다.

데이터를 유지하려면 컨테이너의 호스트 파일 시스템 콘텐츠를 **--volume (-v)** 옵션과 함께 사용하면 됩니다. 컨테이너에 이 볼륨 유형을 사용할 때는 파일 시스템 수준의 권한을 고려해야 합니다.

MariaDB 컨테이너 이미지에서 **mysql** 사용자는 MariaDB가 호스트 시스템에서 실행되고 있는 것처럼 동일한 **/var/lib/mysql** 디렉터리를 소유해야 합니다. 컨테이너에 마운트하려는 디렉터리에는 사용자 및 그룹 소유자로 **mysql**이 있어야 합니다(또는 MariaDB가 호스트 시스템에 설치되지 않은 경우 **mysql** 사용자의 UID 및 GID). 컨테이너를 **root** 사용자로 실행하면 호스트 시스템의 UID 및 GID가 컨테이너 내부의 UID 및 GID와 일치합니다.

UID 및 GID가 일치하는 구성은 rootless 컨테이너에서 동일한 방식으로 발생하지 않습니다. rootless 컨테이너에서는 Podman이 사용자 네임스페이스 내에서 컨테이너를 시작하기 때문에 사용자는 컨테이너 내에서 root 액세스 권한을 갖습니다.

podman unshare 명령을 사용하여 사용자 네임스페이스 내에서 명령을 실행할 수 있습니다. 사용자 네임스페이스에 대한 UID 매핑을 가져오려면 **podman unshare cat** 명령을 사용합니다.

```
[user@host ~]$ podman unshare cat /proc/self/uid_map
0      1000      1
1      100000    65536
[user@host ~]$ podman unshare cat /proc/self/gid_map
0      1000      1
1      100000    65536
```

16장 | 컨테이너 실행

위 출력은 컨테이너에서 root 사용자(UID 및 GID 0)가 호스트 시스템의 사용자(UID 및 GID **1000**)에 매핑됨을 보여줍니다. 컨테이너에서 UID 및 GID 1은 호스트 시스템의 UID 및 GID 100000에 매핑됩니다. 1 이후의 모든 UID 및 GID는 1씩 증가합니다. 예를 들어 컨테이너 내부의 UID 및 GID가 30인 경우 호스트 시스템의 UID 및 GID 100029에 매핑됩니다.

임시 스토리지로 실행 중인 컨테이너 내부의 **mysql** 사용자 UID 및 GID를 보려면 **podman exec** 명령을 사용합니다.

```
[user@host ~]$ podman exec -it db01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
```

/home/user/db_data 디렉터리를 **db01** 컨테이너에 마운트하여 컨테이너의 **/var/lib/mysql** 디렉터리에 영구저장장을 제공하기로 결정합니다. 그런 다음 **/home/user/db_data** 디렉터리를 생성하고 **podman unshare** 명령을 사용하여 사용자 네임스페이스 UID 및 GID **27** 을 디렉터리 소유자로 설정합니다.

```
[user@host ~]$ mkdir /home/user/db_data
[user@host ~]$ podman unshare chown 27:27 /home/user/db_data
```

컨테이너의 UID 및 GID **27** 은 호스트 시스템의 UID 및 GID **100026**에 매핑됩니다. **ls** 명령으로 **/home/user/db_data** 디렉터리의 소유권을 확인하여 매핑을 확인할 수 있습니다.

```
[user@host ~]$ ls -l /home/user/
total 0
drwxrwxr-x. 3 100026 100026 18 May  5 14:37 db_data
...output omitted...
```

이제 올바른 파일 시스템 수준 권한이 설정되었으므로 **podman run** 명령 **-v** 옵션을 사용하여 디렉터리를 마운트합니다.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql \
registry.lab.example.com/rhel8/mariadb-105
```

db01 컨테이너가 실행되고 있지 않는 것을 확인합니다.

CONTAINER ID	IMAGE	COMMAND	
CREATED	STATUS	PORTS	NAMES
dfdc20cf9a7e	registry.lab.example.com/rhel8/mariadb-105:latest	run-mysqld	db01
29 seconds ago	Exited (1)	29 seconds ago	

podman container logs 명령은 **/var/lib/mysql/db_data** 디렉터리에 대한 권한 오류를 표시합니다.

```
[user@host ~]$ podman container logs db01
...output omitted...
--> 16:41:25      Initializing database ...
--> 16:41:25      Running mysql_install_db ...
mkdir: cannot create directory '/var/lib/mysql/db_data': Permission denied
Fatal error Can't create database directory '/var/lib/mysql/db_data'
```

이 오류는 호스트 시스템의 `/home/user/db_data` 디렉터리에 잘못된 SELinux 컨텍스트가 설정되어 있어 발생합니다.

컨테이너 스토리지에 대한 SELinux 컨텍스트

디렉터리를 컨테이너에 영구저장장치로 마운트하려면 먼저 `container_file_t` SELinux 컨텍스트 유형을 설정해야 합니다. 디렉터리에 `container_file_t` SELinux 컨텍스트가 없으면 컨테이너에서 디렉터리에 액세스할 수 없습니다. `Z` 옵션을 `podman run` 명령 `-v` 옵션의 인수에 추가하여 디렉터리에 SELinux 컨텍스트를 자동으로 설정할 수 있습니다.

따라서 `/home/user/db_data` 디렉터리를 `/var/lib/mysql` 디렉터리를 위한 영구저장장치로 마운트할 때 `podman run -v /home/user/db_data:/var/lib/mysql:Z` 명령을 사용하여 해당 디렉터리의 SELinux 컨텍스트를 설정합니다.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
registry.lab.example.com/rhel8/mariadb-105
```

그런 다음 `ls` 명령 `-Z` 옵션을 사용하여 `/home/user/db_data` 디렉터리에 올바른 SELinux 컨텍스트가 설정되어 있는지 확인합니다.

```
[user@host ~]$ ls -Z /home/user/
system_u:object_r:container_file_t:s0:c81,c1009 db_data
...output omitted...
```

컨테이너에 포트 매팅 할당

네트워크를 통해 컨테이너에 액세스할 수 있게 하려면 네트워크 트래픽을 컨테이너의 포트로 전달하는 컨테이너 호스트의 포트에 클라이언트를 연결해야 합니다. 컨테이너 호스트의 네트워크 포트를 컨테이너의 포트에 매팅하면 호스트 네트워크 포트로 전송된 네트워크 트래픽을 컨테이너가 수신하게 됩니다.

예를 들어 MariaDB 컨테이너와 통신하기 위해 컨테이너 호스트의 13306 포트를 컨테이너의 3306 포트에 매팅할 수 있습니다. 따라서 컨테이너 호스트 13306 포트로 전송되는 트래픽을 컨테이너에서 실행 중인 MariaDB가 수신합니다.

컨테이너 호스트의 13306 포트에서 db01 컨테이너의 3306 포트로의 포트 매팅을 설정하려면 `podman run` 명령 `-p` 옵션을 사용합니다.

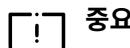
```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
registry.lab.example.com/rhel8/mariadb-105
```

사용 중인 컨테이너 포트 매핑을 모두 표시하려면 `podman port` 명령 `-a` 옵션을 사용합니다. `podman port db01` 명령을 사용하여 `db01` 컨테이너에 매핑된 포트를 표시할 수도 있습니다.

```
[user@host ~]$ podman port -a
1c22fd905120 3306/tcp -> 0.0.0.0:13306
[user@host ~]$ podman port db01
3306/tcp -> 0.0.0.0:13306
```

컨테이너로 리다렉션될 수 있도록 컨테이너 호스트 시스템으로의 포트 **13306** 트래픽을 허용하려면 `firewall-cmd` 명령을 사용합니다.

```
[root@host ~]# firewall-cmd --add-port=13306/tcp --permanent
[root@host ~]# firewall-cmd --reload
```



중요

rootless 컨테이너는 컨테이너의 권한 있는 포트(**1024** 미만)를 열 수 없습니다. 즉, 일반적으로 `podman run -p 80:8080` 명령은 실행 중인 rootless 컨테이너에 대해 작동하지 않습니다. 컨테이너 호스트에서 **1024** 미만의 포트를 컨테이너 포트에 매핑하려면 Podman을 root로 실행하거나 시스템의 다른 사항을 조정해야 합니다.

rootless 컨테이너를 실행 중인 경우에도 컨테이너 호스트에서 **1024** 이상의 포트를 컨테이너의 권한 있는 포트에 매핑할 수 있습니다. **80** 포트에서 수신 대기 중인 서비스를 제공하는 컨테이너에서는 **8080:80** 매핑이 작동합니다.

컨테이너의 DNS 구성

Podman v4.0에서는 컨테이너에 Netavark 및 CNI라는 두 가지 네트워크 백엔드를 지원합니다. RHEL 9부터는 Netavark가 기본적으로 사용됩니다. 사용되는 네트워크 백엔드를 확인하려면 다음 `podman info` 명령을 실행합니다.

```
[user@host ~]$ podman info --format {{.Host.NetworkBackend}}
netavark
```

**참고**

`container-tools` 메타 패키지에는 `netavark` 및 `aardvark-dns` 패키지가 포함됩니다. Podman이 독립 실행형 패키지로 설치되었거나 `container-tools` 메타 패키지가 나중에 설치된 경우, 위 명령의 결과가 `cni`일 수 있습니다. 네트워크 백엔드를 변경하려면 `/usr/share/containers/containers.conf` 파일에서 구성을 다음과 같이 설정하십시오.

```
[network]
...output omitted...
network_backend = "netavark"
```

기본 Podman 네트워크를 사용하는 호스트의 기존 컨테이너에서는 기본 네트워크에 DNS가 활성화되어 있지 않기 때문에 서로의 호스트 이름을 확인할 수 없습니다.

`podman network create` 명령을 사용하여 DNS 사용 네트워크를 생성합니다. `podman network create` 명령을 사용하여 `db_net`이라는 네트워크를 생성하고, 서브넷을 `10.87.0.0/16`으로, 게이트웨이를 `10.87.0.1`로 지정합니다.

```
[user@host ~]$ podman network create --gateway 10.87.0.1 \
--subnet 10.87.0.0/16 db_net
db_net
```

`--gateway` 또는 `--subnet` 옵션을 지정하지 않으면 기본값으로 생성됩니다.

`podman network inspect` 명령은 특정 네트워크에 대한 정보를 표시합니다. `podman network inspect` 명령을 사용하여 게이트웨이와 서브넷이 올바르게 설정되었으며 새 `db_net` 네트워크가 DNS를 사용하는지 확인합니다.

```
[user@host ~]$ podman network inspect db_net
[
  {
    "name": "db_net",
    ...output omitted...
    "subnets": [
      {
        "subnet": "10.87.0.0/16",
        "gateway": "10.87.0.1"
      }
    ],
    ...output omitted...
    "dns_enabled": true,
    ...output omitted...
  ]
]
```

`podman run` 명령 `--network` 옵션을 사용하여 DNS 사용 `db_net` 네트워크를 새 컨테이너에 추가 할 수 있습니다. `podman run` 명령 `--network` 옵션을 사용하여 `db_net` 네트워크에 연결된 `db01` 및 `client01` 컨테이너를 생성합니다.

```
[user@host ~]$ podman run -d --name db01 \
-e MYSQL_USER=student \
-e MYSQL_PASSWORD=student \
```

16장 | 컨테이너 실행

```
-e MYSQL_DATABASE=dev_data \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/user/db_data:/var/lib/mysql:Z \
-p 13306:3306 \
--network db_net \
registry.lab.example.com/rhel8/mariadb-105
[user@host ~]$ podman run -d --name client01 \
--network db_net \
registry.lab.example.com/ubi8/ubi:latest \
sleep infinity
```

컨테이너는 필요한 최소 패키지만 포함하도록 설계되었으므로 컨테이너에 **ping** 및 **ip** 명령 등 통신을 테스트하는데 필요한 유ти리티가 없을 수 있습니다. **podman exec** 명령을 사용하여 컨테이너에 이러한 유ти리를 설치할 수 있습니다.

```
[user@host ~]$ podman exec -it db01 dnf install -y iputils iproute
...output omitted...
[user@host ~]$ podman exec -it client01 dnf install -y iputils iproute
...output omitted...
```

이제 컨테이너에서 컨테이너 이름으로 서로 ping할 수 있습니다. **podman exec** 명령을 사용하여 DNS 확인을 테스트합니다. 이름은 **db_net** 네트워크에 수동으로 설정된 서브넷 내의 IP로 확인됩니다.

```
[user@host ~]$ podman exec -it db01 ping -c3 client01
PING client01.dns.podman (10.87.0.4) 56(84) bytes of data.
64 bytes from 10.87.0.4 (10.87.0.4): icmp_seq=1 ttl=64 time=0.049 ms
...output omitted...
--- client01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 0.049/0.060/0.072/0.013 ms

[user@host ~]$ podman exec -it client01 ping -c3 db01
PING db01.dns.podman (10.87.0.3) 56(84) bytes of data.
64 bytes from 10.87.0.3 (10.87.0.3): icmp_seq=1 ttl=64 time=0.021 ms
...output omitted...
--- db01.dns.podman ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2047ms
rtt min/avg/max/mdev = 0.021/0.040/0.050/0.013 ms
```

podman exec 명령을 사용하여 각 컨테이너의 IP 주소가 DNS 확인과 일치하는지 확인합니다.

```
[user@host ~]$ podman exec -it db01 ip a | grep 10.8
    inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
        inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
[user@host ~]$ podman exec -it client01 ip a | grep 10.8
    inet 10.87.0.3/16 brd 10.87.255.255 scope global eth0
        inet 10.87.0.4/16 brd 10.87.255.255 scope global eth0
```

단일 컨테이너에 대한 여러 네트워크

여러 네트워크를 동시에 컨테이너에 연결하여 다양한 유형의 트래픽을 분리할 수 있습니다.

`podman network create` 명령을 사용하여 backend 네트워크를 생성합니다.

```
[user@host ~]$ podman network create backend
```

그런 다음 `podman network ls` 명령을 사용하여 모든 Podman 네트워크를 봅니다.

```
[user@host ~]$ podman network ls
NETWORK ID      NAME      DRIVER
a7fea510a6d1   backend   bridge
fe680efc5276   db01     bridge
2f259bab93aa   podman   bridge
```

서브넷 및 게이트웨이가 `podman network create` 명령 `--gateway` 및 `--subnet` 옵션을 사용하여 지정되지 않았습니다.

`podman network inspect` 명령을 사용하여 `backend` 네트워크의 IP 정보를 가져옵니다.

```
[user@host ~]$ podman network inspect backend
[
  {
    "name": "backend",
    ...output omitted...
    "subnets": [
      {
        "subnet": "10.89.1.0/24",
        "gateway": "10.89.1.1"
    ...output omitted...
```

컨테이너가 실행 중인 경우 `podman network connect` 명령을 사용하여 컨테이너에 추가 네트워크를 연결할 수 있습니다. `backend` 네트워크를 `db01` 및 `client01` 컨테이너에 연결하려면 `podman network connect` 명령을 사용합니다.

```
[user@host ~]$ podman network connect backend db01
[user@host ~]$ podman network connect backend client01
```



중요

`podman run` 명령을 사용하여 네트워크를 지정하지 않으면 컨테이너가 기본 네트워크에 연결됩니다. 기본 네트워크는 `slirp4netns` 네트워크 모드를 사용하고, `podman network create` 명령으로 생성하는 네트워크는 브리지 네트워크 모드를 사용합니다. `slirp4netns` 네트워크 모드를 사용하여 브리지 네트워크를 컨테이너에 연결하려고 하면 명령이 실패합니다.

```
Error: "slirp4netns" is not supported: invalid network mode
```

`podman inspect` 명령을 사용하여 두 네트워크 모두 각 컨테이너에 연결되어 있는지 확인하고 IP 정보를 표시합니다.

```
[user@host ~]$ podman inspect db01
...output omitted...
    "backend": {
        "EndpointID": "",
        "Gateway": "10.89.1.1",
        "IPAddress": "10.89.1.4",
    },
    "db_net": {
        "EndpointID": "",
        "Gateway": "10.87.0.1",
        "IPAddress": "10.87.0.3",
    }
...output omitted...
[user@host ~]$ podman inspect client01
...output omitted...
    "backend": {
        "EndpointID": "",
        "Gateway": "10.89.1.1",
        "IPAddress": "10.89.1.5",
    },
    "db_net": {
        "EndpointID": "",
        "Gateway": "10.87.0.1",
        "IPAddress": "10.87.0.4",
    }
...output omitted...
```

이제 **client01** 컨테이너가 두 네트워크 모두에서 **db01** 컨테이너와 통신할 수 있습니다. **podman exec** 명령을 사용하여 **client01** 컨테이너의 **db01** 컨테이너에 있는 두 네트워크를 모두 ping합니다.

```
[user@host ~]$ podman exec -it client01 ping -c3 10.89.1.4 | grep 'packet loss'
3 packets transmitted, 3 received, 0% packet loss, time 2052ms
[user@host ~]$ podman exec -it client01 ping -c3 10.87.0.3 | grep 'packet loss'
3 packets transmitted, 3 received, 0% packet loss, time 2054ms
```



참조

podman(1), **podman-exec(1)**, **podman-info(1)**, **podman-network(1)**, **podman-network-create(1)**, **podman-network-inspect(1)**, **podman-network-ls(1)**, **podman-port(1)**, **podman-run(1)** 및 **podman-unshare(1)** 도움말 페이지

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/assembly_working-with-containers_building-running-and-managing-containers

에서 Working with Containers 가이드의 Building, Running, and Managing Linux Containers on Red Hat Enterprise Linux 9장을 참조하십시오.

▶ 연습 가이드

컨테이너 스토리지 및 네트워크 리소스 관리

이 연습에서는 컨테이너를 생성하는 동안 환경 변수를 컨테이너에 전달하고, 영구저장장치를 컨테이너에 마운트하고, 여러 컨테이너 네트워크를 생성 및 연결하고, 호스트 시스템의 컨테이너 포트를 노출합니다.

결과

- 컨테이너 네트워크를 생성하고 컨테이너에 연결합니다.
- 실패한 컨테이너의 문제를 해결합니다.
- 생성하는 동안 환경 변수를 컨테이너에 전달합니다.
- 영구저장장치를 생성하고 컨테이너에 마운트합니다.
- 호스트 포트를 컨테이너 내부의 포트에 매핑합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start containers-resources
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **frontend** 컨테이너 네트워크를 생성합니다. **db_client** 및 **db_01** 컨테이너를 생성하고 **frontend** 네트워크에 연결합니다.

- 2.1. **podman network create** 명령 **--subnet** 및 **--gateway** 옵션을 사용하여 **10.89.1.0/24** 서브넷 및 **10.89.1.1** 게이트웨이가 있는 **frontend** 네트워크를 생성합니다.

```
[student@servera ~]$ podman network create --subnet 10.89.1.0/24 \
--gateway 10.89.1.1 frontend
frontend
```

- 2.2. **registry.lab.example.com** 레지스트리에 로그인합니다.

```
[student@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- 2.3. 백그라운드에서 **db_client**라는 컨테이너를 시작하고 **frontend** 네트워크에 연결합니다. **db_client** 컨테이너에 패키지를 설치할 수 있도록 **/etc/yum.repos.d** 컨테이너 경로에 **/etc/yum.repos.d** DNF 리포지토리 디렉터리를 마운트합니다. **db_client** 컨테이너에서 **sleep infinity** 명령을 실행하여 컨테이너가 종료되지 않도록 합니다. **registry.lab.example.com/ubi9-beta/ubi** 이미지를 사용합니다.

```
[student@servera ~]$ podman run -d --name db_client \
--network frontend \
-v /etc/yum.repos.d:/etc/yum.repos.d \
registry.lab.example.com/ubi9-beta/ubi \
sleep infinity
e20dfed7e392abe4b7bea3c25e9cb17ef95d16af9cedd50d68f997a663ba6c15
```

- 2.4. 백그라운드에서 **db_01**이라는 컨테이너를 시작하고 **frontend** 네트워크에 연결합니다. **registry.lab.example.com/rhel8/mariadb-105** 이미지를 사용합니다.

```
[student@servera ~]$ podman run -d --name db_01 --network frontend \
registry.lab.example.com/rhel8/mariadb-105
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
```

- 2.5. 모든 컨테이너를 봅니다.

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS NAMES
e20dfed7e392	registry.lab.example.com/ubi8/ubi:latest	sleep infinity
56 seconds ago	Up 56 seconds ago	db_client
3e767ae6eea4	registry.lab.example.com/rhel8/mariadb-105:latest	run-mysqld 1 second ago Exited (1) 1 second ago db_01

- ▶ 3. **db_01** 컨테이너의 문제를 해결하고 실행되지 않는 이유를 확인합니다. 필수 환경 변수를 사용하여 **db_01** 컨테이너를 다시 생성합니다.

- 3.1. 컨테이너 로그를 보고 컨테이너가 종료된 이유를 확인합니다.

```
[student@servera ~]$ podman container logs db_01
...output omitted...
You must either specify the following environment variables:
  MYSQL_USER (regex: '^[_a-zA-Z0-9_-]+$')
  MYSQL_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()=-<>,_.?;:|]+$')
  MYSQL_DATABASE (regex: '^[_a-zA-Z0-9_-]+$')
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[_a-zA-Z0-9_-!@#$%^&*()=-<>,_.?;:|]+$')
Or both.
...output omitted...
```

- 3.2. **db_01** 컨테이너를 제거하고 환경 변수를 사용하여 다시 생성합니다. 필요한 모든 환경 변수를 제공합니다.

```
[student@servera ~]$ podman rm db_01
3e767ae6eea4578152a216beb5ae98c8ef03a2d66098debe2736b8b458bab405
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-105
948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
```

- 3.3. 현재 실행 중인 컨테이너를 확인합니다.

CONTAINER ID	IMAGE	COMMAND
CREATED	STATUS	PORTS NAMES
e20dfed7e392	registry.lab.example.com/ubi8/ubi:latest	sleep infinity
56 seconds ago	Up 56 seconds ago	db_client
948c4cd767b5	registry.lab.example.com/rhel8/mariadb-105:latest	run-mysqld
11 seconds ago	Up 12 seconds ago	db_01

- ▶ 4. 컨테이너화된 MariaDB 서비스에 대한 영구저장장치를 생성한 후 로컬 시스템 13306 포트를 컨테이너의 3306 포트에 매핑합니다. **servera** 시스템의 13306 포트에 대한 트래픽을 허용합니다.

- 4.1. **servera** 시스템에서 **/home/student/databases** 디렉터리를 생성합니다.

```
[student@servera ~]$ mkdir /home/student/databases
```

- 4.2. **db_01** 컨테이너에서 mysql UID 및 GID를 가져온 다음 **db01** 컨테이너를 제거합니다.

```
[student@servera ~]$ podman exec -it db_01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
[student@servera ~]$ podman stop db_01
db_01
[student@servera ~]$ podman rm db_01
948c4cd767b561432056e77adb261ab4024c1b66a22af17861aba0f16c66273b
```

- 4.3. 컨테이너 네임스페이스 내에서 chown 명령을 실행하고 **/home/student/database** 디렉터리에서 사용자 및 그룹 소유자를 27로 설정합니다.

```
[student@servera ~]$ podman unshare chown 27:27 /home/student/databases/
[student@servera ~]$ ls -l /home/student/
total 0
drwxr-xr-x. 2 100026 100026 6 May  9 17:40 databases
```

- 4.4. **db_01** 컨테이너를 생성하고 **servera** 시스템의 **/home/student/databases** 디렉터리를 **db_01** 컨테이너 내부의 **/var/lib/mysql** 디렉터리에 마운트합니다. Z 옵션을 사용하여 필수 SELinux 컨텍스트를 적용합니다.

```
[student@servera ~]$ podman run -d --name db_01 \
--network frontend \
-e MYSQL_USER=dev1 \
-e MYSQL_PASSWORD=devpass \
-e MYSQL_DATABASE=devdb \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/student/databases:/var/lib/mysql:z \
-p 13306:3306 \
registry.lab.example.com/rhel8/mariadb-105
```

- 4.5. `db_client` 컨테이너에 `mariadb` 패키지를 설치합니다.

```
[student@servera ~]$ podman exec -it db_client dnf install -y mariadb
...output omitted...
Complete!
```

- 4.6. `db_client` 컨테이너의 `db_01` 컨테이너에 있는 `dev_db` 데이터베이스에 `crucial_data` 표를 생성합니다.

```
[student@servera ~]$ podman exec -it db_client mysql -u dev1 -p -h db_01
Enter password: devpass
...output omitted...
MariaDB [(none)]> USE devdb;
Database changed
MariaDB [devdb]> CREATE TABLE crucial_data(column1 int);
Query OK, 0 rows affected (0.036 sec)

MariaDB [devdb]> SHOW TABLES;
+-----+
| Tables_in_devdb |
+-----+
| crucial_data    |
+-----+
1 row in set (0.001 sec)

MariaDB [devdb]> quit
Bye
```

- 4.7. `servera` 시스템의 방화벽에서 13306 포트의 트래픽을 허용합니다.

```
[student@servera ~]$ sudo firewall-cmd --add-port=13306/tcp --permanent
[sudo] password for student:
success
[student@servera ~]$ sudo firewall-cmd --reload
success
```

- 4.8. `workstation` 시스템에서 두 번째 터미널을 열고, MariaDB 클라이언트를 사용하여 포트 13306에서 `servera` 시스템에 연결하여 영구저장장치에 저장된 `db_01` 컨테이너 내부의 테이블을 표시합니다.

```
[student@workstation ~]$ mysql -u dev1 -p -h servera --port 13306 \
devdb -e 'SHOW TABLES';
Enter password: devpass
+-----+
| Tables_in_devdb |
+-----+
| crucial_data     |
+-----+
```

- ▶ 5. **backend**라는 두 번째 컨테이너 네트워크를 생성한 후 **backend** 네트워크를 **db_client** 및 **db_01** 컨테이너에 연결합니다. 컨테이너 간 네트워크 연결 및 DNS 확인을 테스트합니다.

- 5.1. **10.90.0.0/24** 서브넷 및 **10.90.0.1** 게이트웨이가 있는 **backend** 네트워크를 생성합니다.

```
[student@servera ~]$ podman network create --subnet 10.90.0.0/24 \
--gateway 10.90.0.1 backend
backend
```

- 5.2. **backend** 컨테이너 네트워크를 **db_client** 및 **db_01** 컨테이너에 연결합니다.

```
[student@servera ~]$ podman network connect backend db_client
[student@servera ~]$ podman network connect backend db_01
```

- 5.3. **db_01** 컨테이너의 IP 주소를 가져옵니다.

```
[student@servera ~]$ podman inspect db_01
...output omitted...
{
  "Networks": {
    "backend": {
      "EndpointID": "",
      "Gateway": "10.90.0.1",
      "IPAddress": "10.90.0.3",
      ...
    },
    "frontend": {
      "EndpointID": "",
      "Gateway": "10.89.1.1",
      "IPAddress": "10.89.1.5",
      ...
    }
  }
}
```

- 5.4. **db_client** 컨테이너에 **iputils** 패키지를 설치합니다.

```
[student@servera ~]$ podman exec -it db_client dnf install -y iputils
...output omitted...
Complete!
```

- 5.5. **db_client** 컨테이너에서 **db_01** 컨테이너 이름을 ping합니다.

```
[student@servera ~]$ podman exec -it db_client ping -c4 db_01
PING db_01.dns.podman (10.90.0.3) 56(84) bytes of data.
...output omitted...
--- db_01.dns.podman ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3048ms
rtt min/avg/max/mdev = 0.043/0.049/0.054/0.004 ms
```

5.6. `servera` 시스템을 종료합니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish containers-resources
```

이것으로 섹션을 완료합니다.

컨테이너를 시스템 서비스로 관리

목표

컨테이너를 **systemd** 서비스로 구성하고 부팅 시 시작되도록 컨테이너 서비스를 구성합니다.

systemd 유닛을 사용하여 소규모 컨테이너 환경 관리

컨테이너를 실행하여 시스템 작업을 완료하거나 일련의 명령 출력을 가져올 수 있습니다. 웹 서버 또는 데이터베이스와 같이 서비스를 무기한 실행하는 컨테이너를 실행할 수도 있습니다. 기존 환경에서는 권한 있는 사용자가 일반적으로 이러한 서비스를 시스템 부팅 시 실행하도록 구성하고, **systemctl** 명령을 사용하여 관리합니다.

일반 사용자는 **systemd** 유닛을 생성하여 rootless 컨테이너를 구성할 수 있습니다. 이 구성을 사용하여 **systemctl** 명령으로 컨테이너를 일반 시스템 서비스로 관리할 수 있습니다.

systemd 유닛을 기반으로 컨테이너를 관리하는 방법은 주로 확장할 필요가 없는 기본 및 소규모 배포에 유용합니다. 컨테이너 기반의 많은 애플리케이션과 서비스를 더 정교하게 확장하고 오케스트레이션하기 위해 Red Hat OpenShift Container Platform과 같은 Kubernetes 기반의 엔터프라이즈 오케스트레이션 플랫폼을 사용할 수 있습니다.

이 강의의 주제를 논의하기 위해 다음과 같은 시나리오를 상상해 보십시오.

시스템 관리자는 **http24** 컨테이너 이미지를 기반으로 하는 **webserver1** 컨테이너를 부팅 시 실행하도록 구성해야 합니다. 또한 웹 서버 콘텐츠를 위해 **/app-artifacts** 디렉터리를 마운트하고, 로컬 시스템의 8080 포트를 컨테이너로 매핑해야 합니다. **systemctl** 명령을 사용하여 시작 및 중지하도록 컨테이너를 구성합니다.

systemd 사용자 서비스 요구 사항

일반 사용자는 **systemctl** 명령을 사용하여 서비스를 활성화할 수 있습니다. 세션(그래픽 인터페이스, 텍스트 콘솔 또는 SSH)을 열면 서비스가 시작되고 마지막 세션을 닫으면 중지됩니다. 이 동작은 시스템이 부팅될 때 시작되고 시스템이 종료될 때 중지되는 시스템 서비스와는 다릅니다.

기본적으로 **useradd** 명령을 사용하여 사용자 계정을 생성하면 시스템은 일반 사용자 ID 범위에서 사용 가능한 다음 ID를 사용합니다. 또한 시스템은 **/etc/subuid** 파일에서 사용자 컨테이너에 대한 ID 범위를 예약합니다. **useradd** 명령 **--system** 옵션을 사용하여 사용자 계정을 생성하면 시스템에서 사용자 컨테이너에 대한 범위를 예약하지 않습니다. 따라서 시스템 계정을 사용하여 rootless 컨테이너를 시작할 수 없습니다.

컨테이너를 관리할 전용 사용자 계정을 생성하기로 결정합니다. **useradd** 명령을 사용하여 **appdev-adm** 사용자를 생성하고 **redhat** 을 암호로 사용합니다.

```
[user@host ~]$ sudo useradd appdev-adm
[user@host ~]$ sudo passwd appdev-adm
Changing password for user appdev-adm.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

그런 다음 **su** 명령을 사용하여 **appdev-adm** 사용자로 전환하고 **podman** 명령을 사용하기 시작합니다.

```
[user@host ~]$ su appdev-adm
Password: redhat
[appdev-adm@host ~]$ podman info
ERRO[0000] XDG_RUNTIME_DIR directory "/run/user/1000" is not owned by the current
user
[appdev-adm@host ~]$
```

Podman은 상태 비저장 유틸리티이며 전체 로그인 세션이 필요합니다. Podman은 SSH 세션 내에서 사용해야 하며 **sudo** 또는 **su** 쉘에서 사용할 수 없습니다. 따라서 **su** 쉘을 종료하고 SSH를 통해 시스템에 로그인합니다.

```
[appdev-adm@host ~]$ exit
[user@host ~]$ exit
[user@example ~]$ ssh appdev-adm@host
[appdev-adm@host ~]$
```

그런 다음 컨테이너 레지스트리를 구성하고 자격 증명을 사용하여 인증합니다. 다음 명령을 사용하여 **http** 컨테이너를 실행합니다.

```
[appdev-adm@host ~]$ podman run -d --name webserver1 -p 8080:8080 -v \
~/app-artifacts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24
cde4a3d8c9563fd50cc39de8a4873dcf15a7e881ba4548d5646760eae7a35d81
[appdev-adm@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
cde4a3d8c956 registry.access.redhat.com/ubi8/httpd-24:latest /usr/bin/run-
http... 4 seconds ago Up 5 seconds ago 0.0.0.0:8080->8080/tcp webserver1
```



참고

호스트 파일 시스템에서 컨테이너로 마운트하는 디렉터리에 대한 올바른 액세스 권한을 제공해야 합니다. 컨테이너를 실행할 때 오류가 발생하면 **podman container logs** 명령을 사용하여 트러블슈팅할 수 있습니다.

컨테이너에 대한 **systemd** 사용자 파일 생성

systemd 서비스를 **~/.config/systemd/user/** 디렉터리에 수동으로 정의할 수 있습니다. 사용자 서비스의 파일 구문은 시스템 서비스 파일의 구문과 동일합니다. 자세한 내용은 **systemd.unit(5)** 및 **systemd.service(5)** 도움말 페이지를 검토하십시오.

podman generate systemd 명령을 사용하여 기존 컨테이너에 대한 **systemd** 서비스 파일을 생성합니다. **podman generate systemd** 명령은 컨테이너를 모델로 사용하여 구성 파일을 만듭니다.

podman generate systemd 명령 **--new** 옵션은 서비스가 시작되면 컨테이너를 생성하고 서비스가 중지되면 삭제하도록 **systemd** 서비스를 구성하라고 **podman** 유틸리티에 지시합니다.

**중요**

--new 옵션을 사용하지 않으면 **podman** 유ти리티는 기존 컨테이너를 시작하고 중지할 때 컨테이너를 삭제하지 않도록 서비스 유닛 파일을 구성합니다.

podman generate systemd 명령에 --name 옵션을 사용하여 **webserver1** 컨테이너에 대해 모델링 된 **systemd** 서비스 파일을 표시합니다.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1
...output omitted...
ExecStart=/usr/bin/podman start webserver1 ①
ExecStop=/usr/bin/podman stop -t 10 webserver1 ②
ExecStopPost=/usr/bin/podman stop -t 10 webserver1
...output omitted...
```

- ① 시작 시 **systemd** 데몬은 **podman start** 명령을 실행하여 기존 컨테이너를 시작합니다.
- ② 중지 시 **systemd** 데몬은 **podman stop** 명령을 실행하여 컨테이너를 중지합니다. 이 작업에서 **systemd** 데몬은 컨테이너를 삭제하지 않습니다.

그런 다음 이전 명령에 --new 옵션을 추가하여 **systemd** 구성을 비교합니다.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new
...output omitted...
ExecStartPre=/bin/rm -f %t/%n.ctr-id
ExecStart=/usr/bin/podman run --cidfile=%t/%n.ctr-id --cgroups=no-common --rm --
sdnotify=common --replace -d --name webserver1 -p 8080:8080 -v /home/appdev-adm/
app-artifacts:/var/www/html:Z registry.access.redhat.com/ubi8/httpd-24 ①
ExecStop=/usr/bin/podman stop --ignore --cidfile=%t/%n.ctr-id ②
ExecStopPost=/usr/bin/podman rm -f --ignore --cidfile=%t/%n.ctr-id ③
...output omitted...
```

- ① 시작 시 **systemd** 데몬은 **podman run** 명령을 실행하여 새 컨테이너를 생성한 다음 시작합니다. 이 작업에서는 중지 시 컨테이너를 제거하는 **podman run** 명령 --rm 옵션을 사용합니다.
- ② 중지 시 **systemd**는 **podman stop** 명령을 실행하여 컨테이너를 중지합니다.
- ③ **systemd**는 컨테이너를 중지한 뒤 **systemd**는 **podman rm -f** 명령을 사용하여 제거합니다.

podman generate systemd 명령의 출력을 확인하고 --files 옵션과 함께 위 명령을 실행하여 현재 디렉터리에 **systemd** 사용자 파일을 생성합니다. **webserver1** 컨테이너는 영구저장장치를 사용 하므로 **podman generate systemd** 명령을 --new 옵션과 함께 사용하도록 선택합니다. 그런 다음 ~/.config/systemd/user/ 디렉터리를 생성하고 파일을 이 위치로 이동합니다.

```
[appdev-adm@host ~]$ podman generate systemd --name webserver1 --new --files
/home/appdev-adm/container-webserver1.service
[appdev-adm@host ~]$ mkdir -p ~/.config/systemd/user/
[appdev-adm@host ~]$ mv container-webserver1.service ~/.config/systemd/user/
```

컨테이너에 대한 **systemd** 사용자 파일 관리

이제 **systemd** 사용자 파일을 생성했으므로 **systemctl** 명령 **--user** 옵션을 사용하여 **webserver1** 컨테이너를 관리할 수 있습니다.

먼저 **systemctl** 명령에서 새 사용자 파일을 인식하도록 **systemd** 데몬을 다시 로드합니다. **systemctl --user start** 명령을 사용하여 **webserver1** 컨테이너를 시작합니다. 컨테이너에 대해 생성된 **systemd** 사용자 파일의 이름을 사용합니다.

```
[appdev-adm@host ~]$ systemctl --user daemon-reload
[appdev-adm@host ~]$ systemctl --user start container-webserver1.service
[appdev-adm@host ~]$ systemctl --user status container-webserver1.service
● container-webserver1.service - Podman container-webserver1.service
   Loaded: loaded (/home/appdev-adm/.config/systemd/user/container-
webserver1.service; disabled; vendor preset: disabled)
     Active: active (running) since Thu 2022-04-28 21:22:26 EDT; 18s ago
       Docs: man:podman-generate-systemd(1)
    Process: 31560 ExecStartPre=/bin/rm -f /run/user/1003/container-
webserver1.service.ctr-id (code=exited, status=0/SUCCESS)
      Main PID: 31600 (common)
     ...output omitted...
[appdev-adm@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
18eb00f42324 registry.access.redhat.com/ubi8/httpd-24:latest /usr/bin/run-
http... 28 seconds ago Up 29 seconds ago 0.0.0.0:8080->8080/tcp webserver1
Created symlink /home/appdev-adm/.config/systemd/user/default.target.wants/
container-webserver1.service → /home/appdev-adm/.config/systemd/user/container-
webserver1.service.
```



중요

systemd 데몬을 사용하여 컨테이너를 구성하는 경우 데몬이 컨테이너 상태를 모니터링하고 실패하는 경우 컨테이너를 다시 시작합니다. **podman** 명령을 사용하여 이러한 컨테이너를 시작하거나 중지하지 마십시오. 그러면 **systemd** 데몬 모니터링에 방해가 될 수 있습니다.

아래 표에는 **systemd** 시스템 서비스와 사용자 서비스 간에 사용되는 디렉터리와 명령이 요약되어 있습니다.

시스템 서비스와 사용자 서비스 비교

사용자 지정 유닛 파일 저장	시스템 서비스	/etc/systemd/system/unit.service
	사용자 서비스	~/.config/systemd/user/unit.service
유닛 파일 다시 로드	시스템 서비스	# systemctl daemon-reload
	사용자 서비스	\$ systemctl --user daemon-reload

서비스 시작 및 중지	시스템 서비스	<code># systemctl start UNIT</code> <code># systemctl stop UNIT</code>
	사용자 서비스	<code>\$ systemctl --user start UNIT</code> <code>\$ systemctl --user stop UNIT</code>
시스템이 시작될 때 서비스 시작	시스템 서비스	<code># systemctl enable UNIT</code>
	사용자 서비스	<code>\$ logindctl enable-linger</code> <code>\$ systemctl --user enable UNIT</code>

시스템 부팅 시 시작할 컨테이너 구성

이제 `systemd` 서비스 구성이 지정된 사용자의 컨테이너를 실행할 준비가 되었습니다. 그러나 `systemd` 서비스는 사용자가 시스템에서 로그아웃하면 일정 시간 후에 컨테이너를 중지합니다. 이러한 동작은 `systemd` 서비스 유닛을 사용자 로그인 시 서비스를 시작하고 사용자 로그아웃 시 중지하는 `--user` 옵션을 사용하여 생성했기 때문에 발생합니다.

`logindctl enable-linger` 명령을 실행하여 이 기본 동작을 변경하고 활성화된 서비스가 서버와 함께 시작되고 서버가 종료되면 중지되도록 강제 실행할 수 있습니다.

`logindctl` 명령을 사용하여 구성된 서비스의 마지막 사용자 세션이 종료된 후에도 유지되도록 `systemd` 사용자 서비스를 구성합니다. 그런 다음 `logindctl show-user` 명령을 사용하여 성공적으로 구성되었는지 확인합니다.

```
[user@host ~]$ logindctl show-user appdev-adm
...output omitted...
Linger=no
[user@host ~]$ logindctl enable-linger
[user@host ~]$ logindctl show-user appdev-adm
...output omitted...
Linger=yes
```

작업을 되돌리려면 `logindctl disable-linger` 명령을 사용합니다.

systemd를 사용하여 루트로 컨테이너 관리

루트로 실행하고 `systemd` 서비스 파일을 사용하여 관리하도록 컨테이너를 구성할 수도 있습니다. 이 접근 방식의 한 가지 이점은 특정 사용자로 작동하는 것이 아니라 일반 `systemd` 유닛 파일과 똑같은 방식으로 작동하도록 해당 서비스 파일을 구성할 수 있다는 것입니다.

서비스 파일을 root로 설정하는 절차는 다음을 제외하고 rootless 컨테이너에 대해 이전에 설명한 절차와 유사합니다.

- 컨테이너 관리를 위해 전용 사용자를 생성하지 마십시오.
- 서비스 파일은 `~/.config/systemd/user` 디렉터리가 아닌 `/etc/systemd/system` 디렉터리에 있어야 합니다.
- `--user` 옵션 없이 `systemctl` 명령을 사용하여 컨테이너를 관리합니다.

- root 사용자로 `logindctl enable-linger` 명령을 실행하지 마십시오.

시연을 보려면 이 섹션 마지막 부분의 참조에 나열된 Red Hat Videos 채널의 YouTube 동영상을 참조하십시오.



참조

`logindctl(1)`, `systemd.unit(5)`, `systemd.service(5)`, `subuid(5)` 및 `podman-generate-systemd(1)` 도움말 페이지

Systemd 유닛 파일을 사용하여 Podman에서 컨테이너 관리

<https://www.youtube.com/watch?v=AGkM2jGT61Y>

자세한 내용은

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/building_running_and_managing_containers/index

에서 Running Containers as Systemd Services with Podman 가이드의 Red Hat Enterprise Linux 9 Building, Running, and Managing Containers 장을 참조하십시오.

▶ 연습 가이드

컨테이너를 시스템 서비스로 관리

이 연습에서는 **systemd** 서비스로 관리할 컨테이너를 구성하고, **systemctl** 명령을 사용하여 호스트 시스템이 시작될 때 자동으로 시작되도록 해당 컨테이너를 관리합니다.

결과

- 컨테이너를 관리할 **systemd** 서비스 파일을 생성합니다.
- 컨테이너를 **systemctl** 명령으로 관리할 수 있도록 구성합니다.
- 호스트 시스템이 시작될 때 **systemd** 사용자 서비스에서 컨테이너를 시작하도록 사용자 계정을 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start containers-services
```

지침

- ▶ 1. **student** 사용자로 **servera** 시스템에 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- ▶ 2. **redhat**을 암호로 사용하여 **contsvc**라는 사용자 계정을 생성합니다. 이 사용자 계정을 사용하여 컨테이너를 **systemd** 서비스로 실행합니다.

- 2.1. **contsvc** 사용자를 생성합니다. **contsvc** 사용자의 암호를 **redhat**으로 설정합니다.

```
[student@servera ~]$ sudo useradd contsvc
[sudo] password for student: student
[student@servera ~]$ sudo passwd contsvc
Changing password for user contsvc.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 2.2. **systemd** 계정으로 **contsvc** 사용자 서비스를 관리하려면 **contsvc** 사용자로 직접 로그인해야 합니다. **su** 및 **sudo** 명령을 사용하여 **contsvc** 사용자가 포함된 세션을 생성할 수 있습니다.

student 사용자로 **workstation** 시스템으로 돌아간 다음 **contsvc** 사용자로 로그인합니다.

```
[student@servera ~]$ exit
logout
Connection to servera closed.
[student@workstation ~]$ ssh contsvc@servera
...output omitted...
[contsvc@servera ~]$
```

- ▶ 3. 훈 디렉터리의 **registry.lab.example.com** 강의실 레지스트리에 대한 액세스를 구성합니다. **/tmp/containers-services/registries.conf** 파일을 템플릿으로 사용합니다.

- 3.1. **~/.config/containers/** 디렉터리를 생성합니다.

```
[contsvc@servera ~]$ mkdir -p ~/.config/containers/
```

- 3.2. **lab** 스크립트를 통해 **/tmp/containers-services/** 디렉터리에 **registries.conf** 파일을 준비합니다. 해당 파일을 **~/.config/containers/** 디렉터리에 복사합니다.

```
[contsvc@servera ~]$ cp /tmp/containers-services/registries.conf \
~/.config/containers/
```

- 3.3. **registry.lab.example.com** 레지스트리에 액세스할 수 있는지 확인합니다. 모두 정상 작동하는 경우 이 명령은 일부 이미지를 나열해야 합니다.

```
[contsvc@servera ~]$ podman search ubi
NAME                                     DESCRIPTION
registry.lab.example.com/ubi7/ubi
registry.lab.example.com/ubi8/ubi
registry.lab.example.com/ubi9-beta/ubi
```

- ▶ 4. **/home/contsvc/webcontent/html/** 디렉터리를 웹 서버 컨테이너의 영구저장장치로 사용합니다. 디렉터리 내부에 **Hello World** 행을 사용하여 **index.html** 테스트 페이지를 생성합니다.

- 4.1. **~/webcontent/html/** 디렉터리를 생성합니다.

```
[contsvc@servera ~]$ mkdir -p ~/webcontent/html/
```

- 4.2. **index.html** 파일을 생성하고 **Hello World** 행을 추가합니다.

```
[contsvc@servera ~]$ echo "Hello World" > ~/webcontent/html/index.html
```

- 4.3. 다른 사람에 대한 권한이 **webcontent/html** 디렉터리에서 **r-x**로 설정되어 있고 **index.html** 파일에서 **r--**로 설정되어 있는지 확인합니다. 컨테이너에서는 권한이 없는 사용자를 사용하지만, **index.html** 파일을 읽을 수 있어야 합니다.

```
[contsvc@servera ~]$ ls -ld webcontent/html/
drwxr-xr-x. 2 contsvc contsvc 24 Aug 28 04:56 webcontent/html/
[contsvc@servera ~]$ ls -l webcontent/html/index.html
-rw-r--r--. 1 contsvc contsvc 12 Aug 28 04:56 webcontent/html/index.html
```

- ▶ 5. `registry.lab.example.com/rhel8/httpd-24:1-163` 이미지를 사용하여 분리 모드에서 `webapp`이라는 컨테이너를 실행합니다. 로컬 호스트의 8080 포트를 컨테이너의 8080 포트로 리디렉션합니다. 호스트의 `~/webcontent` 디렉터리를 컨테이너의 `/var/www` 디렉터리에 마운트합니다.

- 5.1. `registry.lab.example.com`을 암호로 사용하여 `admin` 사용자로 `redhat321` 레지스트리에 로그인합니다.

```
[contsvc@servera ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

- 5.2. `registry.lab.example.com/rhel8/httpd-24:1-163` 이미지를 사용하여 분리 모드에서 `webapp`이라는 컨테이너를 실행합니다. `-p` 옵션을 사용하여 `servera`의 8080 포트를 컨테이너의 8080 포트에 매핑합니다. `-v` 옵션을 사용하여 `servera`의 `~/webcontent` 디렉터리를 컨테이너의 `/var/www` 디렉터리에 마운트합니다.

```
[contsvc@servera ~]$ podman run -d --name webapp -p 8080:8080 -v \
~/webcontent:/var/www:Z registry.lab.example.com/rhel8/httpd-24:1-163
750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06
...output omitted...
```

- 5.3. 웹 서비스가 포트 8080에서 작동하는지 확인합니다.

```
[contsvc@servera ~]$ curl http://localhost:8080
Hello World
```

- ▶ 6. `systemctl` 명령으로 `webapp` 컨테이너를 관리하기 위해 `systemd` 서비스 파일을 생성합니다. 서비스를 시작할 때 `systemd` 데몬이 컨테이너를 생성하도록 `systemd` 서비스를 구성합니다. 구성이 완료한 후 `webapp` 컨테이너를 중지한 다음 삭제합니다. `systemd` 데몬은 컨테이너가 초기에 존재하지 않을 것으로 예상합니다.

- 6.1. `~/.config/systemd/user/` 디렉터리를 생성하여 변경합니다.

```
[contsvc@servera ~]$ mkdir -p ~/.config/systemd/user/
[contsvc@servera ~]$ cd ~/.config/systemd/user
```

- 6.2. `webapp` 컨테이너에 대한 유닛 파일을 생성합니다. `systemd` 가 서비스가 시작될 때 컨테이너를 만들고 서비스를 중지할 때 컨테이너를 삭제하도록 `--new` 옵션을 사용합니다.

```
[contsvc@servera user]$ podman generate systemd --name webapp --files --new
/home/contsvc/.config/systemd/user/container-webapp.service
```

- 6.3. `webapp` 컨테이너를 중지한 다음 삭제합니다.

```
[contsvc@servera user]$ podman stop webapp
webapp
[contsvc@servera user]$ podman rm webapp
750a681bd37cb6825907e9be4347eec2c4cd79550439110fc6d41092194d0e06
[contsvc@servera user]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

- ▶ 7. **systemd** 데몬 구성을 다시 로드한 다음 새 **container-webapp** 사용자 서비스를 활성화하고 시작합니다. **systemd** 서비스 구성을 확인하고, 서비스를 중지 및 시작하고, 웹 서버 응답과 컨테이너 상태를 표시합니다.

7.1. 새 유닛 파일을 인식하도록 구성을 다시 로드합니다.

```
[contsvc@servera user]$ systemctl --user daemon-reload
```

7.2. **container-webapp** 서비스를 활성화하고 시작합니다.

```
[contsvc@servera user]$ systemctl --user enable --now container-webapp
Created symlink /home/contsvc/.config/systemd/user/default.target.wants/container-
webapp.service → /home/contsvc/.config/systemd/user/container-webapp.service.
```

7.3. 웹 서버가 요청에 응답하는지 확인합니다.

```
[contsvc@servera user]$ curl http://localhost:8080
Hello World
```

7.4. 컨테이너가 실행되고 있는지 확인합니다.

```
[contsvc@servera user]$ podman ps
CONTAINER ID IMAGE COMMAND
CREATED STATUS PORTS NAMES
3e996db98071 registry.access.redhat.com/ubi8/httpd-24:1-163 /usr/bin/run-http...
3 minutes ago Up 3 minutes ago 0.0.0.0:8080->8080/tcp webapp
```

서비스를 다시 시작할 때 **systemd** 데몬이 컨테이너를 생성하는지 확인하려면 이 컨테이너 ID 정보가 필요합니다.

7.5. **container-webapp** 서비스를 중지한 다음 컨테이너가 더 이상 존재하지 않는 것을 확인합니다. 서비스를 중지하면 **systemd** 데몬이 컨테이너를 중지한 다음 삭제합니다.

```
[contsvc@servera user]$ systemctl --user stop container-webapp
[contsvc@servera user]$ podman ps --all
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
```

7.6. **container-webapp** 서비스를 시작한 다음 컨테이너가 실행 중인지 확인합니다.

systemd 데몬은 시작 명령으로 컨테이너를 만들고 중지 명령으로 컨테이너를 삭제하므로 컨테이너 ID가 다릅니다.

```
[contsvc@servera user]$ systemctl --user start container-webapp
[contsvc@servera user]$ podman ps
CONTAINER ID  IMAGE                               COMMAND
CREATED      STATUS     PORTS                 NAMES
4584b4df514c  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/run-http...
6 seconds ago  Up 7 seconds ago  0.0.0.0:8080->8080/tcp  webapp
```

- ▶ 8. 시스템 부팅 시 **contsvc** 사용자에 대한 서비스가 시작되는지 확인합니다. 완료되면 **servera** 시스템을 다시 시작합니다.

- 8.1. **logindctl enable-linger** 명령을 실행합니다.

```
[contsvc@servera user]$ logindctl enable-linger
```

- 8.2. **Linger** 사용자에 **contsvc** 옵션이 설정되었는지 확인합니다.

```
[contsvc@servera user]$ logindctl show-user contsvc
...output omitted...
Linger=yes
```

- 8.3. **root** 사용자로 전환한 다음, **systemctl reboot** 명령을 사용하여 **servera**를 다시 시작합니다.

```
[contsvc@servera user]$ su -
Password: redhat
Last login: Fri Aug 28 07:43:40 EDT 2020 on pts/0
[root@servera ~]# systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@workstation ~]$
```

- ▶ 9. **servera** 시스템이 다시 작동하면 **servera**에 **contsvc** 사용자로 로그인합니다. **systemd** 데몬이 **webapp** 컨테이너를 시작했고 웹 콘텐츠를 사용할 수 있는지 확인합니다.

- 9.1. **contsvc** 사용자로 **servera**에 로그인합니다.

```
[student@workstation ~]$ ssh contsvc@servera
...output omitted...
```

- 9.2. 컨테이너가 실행되고 있는지 확인합니다.

```
[contsvc@servera ~]$ podman ps
CONTAINER ID  IMAGE                               COMMAND
CREATED      STATUS     PORTS                 NAMES
6c325bf49f84  registry.access.redhat.com/ubi8/httpd-24:1-163  /usr/bin/run-http...
2 minutes ago  Up 2 minutes ago  0.0.0.0:8080->8080/tcp  webapp
```

- 9.3. 웹 콘텐츠에 액세스합니다.

```
[contsvc@servera ~]$ curl http://localhost:8080  
Hello World
```

9.4. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[contsvc@servera ~]$ exit  
logout  
Connection to servera closed.  
[student@workstation ~]$
```

완료

workstation 시스템에서 **lab finish containers-services** 스크립트를 실행하여 이 연습을 완료합니다.

```
[student@workstation ~]$ lab finish containers-services
```

이것으로 세션을 완료합니다.

▶ 랩

컨테이너 실행

이 랩에서는 MariaDB 데이터베이스 서비스를 제공하고 영구저장장치에 데이터베이스를 저장하며 서버와 함께 자동으로 시작되는 컨테이너를 서버에 구성합니다.

결과

- 분리된 컨테이너를 만듭니다.
- 포트 리디렉션 및 영구저장장치를 구성합니다.
- 호스트 시스템이 시작될 때 시작할 컨테이너에 대해 **systemd**를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start containers-review
```

지침

1. **serverb**에서 컨테이너 툴 패키지를 설치합니다.
2. **registry.lab.example.com**의 컨테이너 이미지 레지스트리는 여러 태그를 사용하여 **rhel8/mariadb-103** 이미지를 저장합니다. **podsvc** 사용자를 사용하여 사용 가능한 태그를 나열하고, 가장 낮은 버전 번호와 함께 태그를 적어둡니다. **admin** 사용자 및 **redhat321** 암호를 사용하여 레지스트리에 인증합니다. **/tmp/registries.conf** 파일을 레지스트리 구성 템플릿으로 사용합니다.
3. 컨테이너에 읽기/쓰기 액세스 권한이 있도록 **/home/podsvc/db_data** 디렉터리를 만들고 구성합니다. 그런 다음 분리된 **inventorydb** 컨테이너를 만듭니다. **registry.lab.example.com** 레지스트리의 **rhel8/mariadb-103** 이미지를 사용하고, 이전 단계에서 찾은 해당 이미지에서 가장 낮은 버전 번호를 사용하여 태그를 지정합니다. 컨테이너의 3306 포트를 호스트의 13306 포트에 매핑합니다. 호스트의 **/home/podsvc/db_data** 디렉터리를 컨테이너의 **/var/lib/mysql/data**로 마운트합니다. 컨테이너에 다음 변수 값을 선언합니다.

변수	값
MYSQL_USER	operator1
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

/home/podsvc/containers-review/variables의 **serverb** 파일에서 이 매개 변수를 복사하여 붙여 넣을 수 있습니다. MariaDB 데이터베이스가 실행 중인지 확인하려면 /home/podsvc/containers-review/testdb.sh 스크립트를 실행합니다.

4. 시스템이 부팅될 때 inventorydb 컨테이너가 자동으로 시작되도록 **systemd** 데몬을 구성합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade containers-review
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish containers-review
```

이것으로 섹션을 완료합니다.

▶ 솔루션

컨테이너 실행

이 랩에서는 MariaDB 데이터베이스 서비스를 제공하고 영구저장장치에 데이터베이스를 저장하며 서버와 함께 자동으로 시작되는 컨테이너를 서버에 구성합니다.

결과

- 분리된 컨테이너를 만듭니다.
- 포트 리디렉션 및 영구저장장치를 구성합니다.
- 호스트 시스템이 시작될 때 시작할 컨테이너에 대해 **systemd**를 구성합니다.

시작하기 전에

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start containers-review
```

지침

1. **serverb**에서 컨테이너 툴 패키지를 설치합니다.

- 1.1. **student** 사용자로 **serverb**에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

- 1.2. **container-tools** 패키지를 설치합니다.

```
[student@serverb ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

2. **registry.lab.example.com**의 컨테이너 이미지 레지스트리는 여러 태그를 사용하여 **rhel8/mariadb-103** 이미지를 저장합니다. **pods** 사용자를 사용하여 사용 가능한 태그를 나열하고, 가장 낮은 버전 번호와 함께 태그를 적어 둡니다. **admin** 사용자 및 **redhat321** 암호를 사용하여 레지스트리에 인증합니다. **/tmp/registries.conf** 파일을 레지스트리 구성 템플릿으로 사용합니다.

- 2.1. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

2.2. `podsvc` 사용자로 `serverb`에 로그인합니다.

```
[student@workstation ~]$ ssh podsvc@serverb
...output omitted...
[podsvc@serverb ~]$
```

2.3. 홈 디렉터리의 `registry.lab.example.com` 강의실 레지스트리에 대한 액세스를 구성합니다. `/tmp/registries.conf` 파일을 템플릿으로 사용합니다.

```
[podsvc@serverb ~]$ mkdir -p ~/.config/containers/
[podsvc@serverb ~]$ cp /tmp/registries.conf \
~/.config/containers/
```

2.4. `podman login` 명령을 사용하여 컨테이너 레지스트리에 로그인합니다.

```
[podsvc@serverb ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```



참고

`mariadb` 컨테이너 이미지가 포함된 리포지토리는 공용 리포지토리가 아니므로 `podman search mariadb` 명령은 결과를 반환하지 않습니다. `podman-search` (1) man 페이지에서 이미지 존재 여부를 확인하기 위해 `podman-search`를 사용하는 경우의 비신뢰성에 대한 참고 사항을 검토하시기 바랍니다.

2.5. `registry.lab.example.com/rhel8/mariadb-103` 이미지에 대한 정보를 봅니다.

```
[podsvc@serverb ~]$ skopeo inspect \
docker://registry.lab.example.com/rhel8/mariadb-103
{
  "Name": "registry.lab.example.com/rhel8/mariadb-103",
  "Digest": "sha256:a95b...4816",
  "RepoTags": [
    "1-86",
    "1-102",
    "latest"
  ],
  ...output omitted...
```

가장 낮은 버전 태그는 **1-86** 버전입니다.

3. 컨테이너에 읽기/쓰기 액세스 권한이 있도록 `/home/podsvc/db_data` 디렉터리를 만들고 구성합니다. 그런 다음 분리된 `inventorydb` 컨테이너를 만듭니다. `registry.lab.example.com` 레

지스트리의 **rhel8/mariadb-103** 이미지를 사용하고, 이전 단계에서 찾은 해당 이미지에서 가장 낮은 버전 번호를 사용하여 태그를 지정합니다. 컨테이너의 3306 포트를 호스트의 13306 포트에 매핑합니다. 호스트의 **/home/podsvc/db_data** 디렉터리를 컨테이너의 **/var/lib/mysql/data**로 마운트합니다. 컨테이너에 다음 변수 값을 선언합니다.

변수	값
MYSQL_USER	operator1
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

/home/podsvc/containers-review/variables의 **serverb** 파일에서 이 매개 변수를 복사하여 붙여 넣을 수 있습니다. MariaDB 데이터베이스가 실행 중인지 확인하려면 **/home/podsvc/containers-review/testdb.sh** 스크립트를 실행합니다.

3.1. 분리된 **db_01** 컨테이너를 시작하여 **mysql** UID 및 GID를 가져옵니다.

```
[podsvc@serverb ~]$ podman run -d --name db_01 -p 13306:3306 \
-e MYSQL_USER=operator1 \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
registry.lab.example.com/rhel8/mariadb-103:1-86
...output omitted...
c33f85d177dc8c51a303e231e6be63c1f251b9d426b4ccb56498603ab72d4219
```

3.2. **/home/podsvc/db_data** 디렉터리를 생성합니다.

```
[podsvc@serverb ~]$ mkdir /home/podsvc/db_data
```

3.3. **db_01** 컨테이너에서 **mysql** UID 및 GID를 가져온 다음 **db01** 컨테이너를 제거합니다.

```
[podsvc@serverb ~]$ podman exec -it db_01 grep mysql /etc/passwd
mysql:x:27:27:MySQL Server:/var/lib/mysql:/sbin/nologin
[podsvc@serverb ~]$ podman stop db_01
db_01
[podsvc@serverb ~]$ podman rm db_01
c33f85d177dc8c51a303e231e6be63c1f251b9d426b4ccb56498603ab72d4219
```

3.4. **podman unshare** 명령을 사용하여 사용자 네임스페이스 UID 및 GID 27을 디렉터리 소유자로 설정합니다.

```
[podsvc@serverb ~]$ podman unshare chown 27:27 /home/podsvc/db_data
```

3.5. 컨테이너를 생성합니다.

```
[podsvc@serverb ~]$ podman run -d --name inventorydb -p 13306:3306 \
-e MYSQL_USER=operator1 \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
-v /home/podsvc/db_data:/var/lib/mysql/data:Z \
registry.lab.example.com/rhel8/mariadb-103:1-86
...output omitted...
```

3.6. 데이터베이스가 실행 중인지 확인합니다.

```
[podsvc@serverb ~]$ ~/containers-review/testdb.sh
Testing the access to the database...
SUCCESS
```

4. 시스템이 부팅될 때 **inventorydb** 컨테이너가 자동으로 시작되도록 **systemd** 데몬을 구성합니다.

4.1. **sudo** 또는 **su**를 사용하여 **podsvc** 사용자로 로그인한 경우 **serverb**를 종료하고 **ssh** 명령을 사용하여 **podsvc** 사용자로 직접 **serverb**에 로그인합니다. **systemd** 데몬을 사용하려면 사용자가 콘솔 또는 SSH를 통해 직접 세션을 열어야 합니다. SSH를 사용하여 **podsvc** 사용자로 **serverb** 시스템에 이미 로그인한 경우 이 단계를 생략합니다.

```
[student@workstation ~]$ ssh podsvc@serverb
...output omitted...
[podsvc@serverb ~]$
```

4.2. **~/.config/systemd/user/** 디렉터리를 생성합니다.

```
[podsvc@serverb ~]$ mkdir -p ~/.config/systemd/user/
```

4.3. 실행 중인 컨테이너에서 **systemd** 유닛 파일을 생성합니다.

```
[podsvc@serverb ~]$ cd ~/.config/systemd/user/
[podsvc@serverb user]$ podman generate systemd --name inventorydb --files --new
/home/podsvc/.config/systemd/user/container-inventorydb.service
```

4.4. **inventorydb** 컨테이너를 중지한 다음 삭제합니다.

```
[podsvc@serverb user]$ podman stop inventorydb
inventorydb
[podsvc@serverb user]$ podman rm inventorydb
0d28f0e0a4118ff019691e34afe09b4d28ee526079b58d19f03b324bd04fd545
```

4.5. 구성을 다시 로드하도록 **systemd** 데몬에 지시한 다음 **container-inventorydb** 서비스를 활성화하고 시작합니다.

```
[podsvc@serverb user]$ systemctl --user daemon-reload
[podsvc@serverb user]$ systemctl --user enable --now container-inventorydb.service
Created symlink /home/podsvc/.config/systemd/user/default.target.wants/
container-inventorydb.service → /home/podsvc/.config/systemd/user/container-
inventorydb.service.
```

4.6. 컨테이너가 실행 중인지 확인합니다.

```
[podsvc@serverb user]$ ~/containers-review/testdb.sh
Testing the access to the database...
SUCCESS
[podsvc@serverb user]$ podman ps
CONTAINER ID  IMAGE                                COMMAND      CREATED
           STATUS     PORTS          NAMES
3ab24e7f000d  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  47
              seconds ago   Up 46 seconds ago  0.0.0.0:13306->3306/tcp  inventorydb
```

4.7. 서버가 시작될 때 사용자 서비스가 자동으로 시작되도록 `loginctl enable-linger` 명령을 실행합니다.

```
[podsvc@serverb ~]$ loginctl enable-linger
```

4.8. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[podsvc@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade containers-review
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish containers-review
```

이것으로 세션을 완료합니다.

요약

- 컨테이너는 호스트에 설치된 소프트웨어와 충돌하지 않도록 애플리케이션 및 해당 종속성을 간편하게 배포하고 실행할 수 있는 방법입니다.
- 컨테이너 레지스트리에서 컨테이너 이미지를 다운로드하거나 직접 만들어 컨테이너를 실행할 수 있습니다.
- 컨테이너 파일을 명령과 함께 사용하여 사용자 지정 컨테이너 이미지를 빌드할 수 있습니다.
- Red Hat Enterprise Linux에서 제공하는 Podman은 단일 호스트에서 컨테이너와 컨테이너 이미지를 직접 실행하고 관리합니다.
- 보안을 강화하기 위해 컨테이너를 **root**로 실행하거나 권한이 없는 rootless 컨테이너로 실행할 수 있습니다.
- 컨테이너 호스트의 네트워크 포트를 매팅하여 컨테이너에서 실행되는 서비스에 트래픽을 전달할 수 있습니다.
- 환경 변수를 사용하여 빌드 시 컨테이너에 소프트웨어를 구성할 수 있습니다.
- 컨테이너 스토리지는 임시지만 컨테이너 호스트의 디렉터리 콘텐츠를 사용하는 등의 방법으로 영구저장 장치를 컨테이너에 연결할 수 있습니다.
- 시스템이 시작되면 컨테이너를 자동으로 실행하도록 **systemd** 유닛 파일을 구성할 수 있습니다.

종합 검토

목적

RHCSA Rapid Track 교육 과정의 작업을 검토합니다.

목표

- RHCSA Rapid Track 교육 과정의 작업을 검토합니다.

섹션

- 종합 검토

랩

- 부팅 문제 해결 및 서버 유지 관리
- 파일 시스템 및 스토리지 구성 및 관리
- 서버 보안 구성 및 관리
- 컨테이너 실행

종합 검토

목표

이 섹션을 마치면 RHCSA Rapid Track에서 배운 지식과 기술을 검토하고 적용할 수 있습니다.

RHCSA Rapid Track 검토

이 교육 과정을 포괄적으로 검토하기 전에 각 장에서 다룬 주제를 충분히 이해하고 있어야 합니다.

추가 학습을 위해 교과서의 앞 섹션을 참조할 수 있습니다.

1장. 시스템 액세스 및 지원 받기

텍스트 파일을 편집하고, 로컬 및 원격 Linux 시스템에 로그인하며, Red Hat Support 및 Red Hat Insights를 통해 제공되는 문제 해결 방법을 조사합니다.

- vim 편집기를 사용하여 명령 줄에서 텍스트 파일을 생성하고 편집합니다.
- 암호 없이 원격 시스템에 안전하게 로그인하기 위해 키 기반 인증을 사용하도록 사용자 계정을 구성합니다.
- Red Hat Customer Portal 주요 리소스를 설명하고 사용하여 Red Hat 설명서 및 Knowledgebase에서 정보를 찾습니다.
- Red Hat Insights를 사용하여 서버의 문제를 분석하고 문제를 수정하거나 해결한 다음, 솔루션이 성공했는지 확인합니다.

2장. 명령줄에서 파일 관리

Bash 쉘에서 파일을 복사, 이동, 생성, 삭제 및 구성합니다.

- Linux에서 파일을 구성하는 방법 및 파일 시스템 계층 구조에 있는 여러 디렉터리의 용도를 설명합니다.
- 하드 링크와 심볼릭(또는 "소프트") 링크를 사용하여 여러 파일 이름이 동일한 파일을 참조하도록 합니다.
- Bash 쉘의 패턴 일치 기능을 사용하여 많은 파일에 영향을 주는 명령을 효율적으로 실행합니다.

3장. 로컬 사용자 및 그룹 관리

로컬 사용자 및 그룹을 생성, 관리, 삭제하고 로컬 암호 정책을 관리합니다.

- Linux 시스템에 있는 사용자 및 그룹의 목적을 설명합니다.
- 수퍼유저 계정으로 전환하여 Linux 시스템을 관리하고, **sudo** 명령을 통해 다른 사용자에게 수퍼유저 액세스 권한을 부여합니다.
- 로컬 사용자 계정을 생성, 관리 및 삭제합니다.
- 로컬 그룹 계정을 생성, 수정 및 삭제합니다.
- 사용자에 대한 암호 관리 정책을 설정하고, 수동으로 사용자 계정을 잠그거나 잠금을 해제합니다.

4장. 파일에 대한 액세스 제어

파일에 대해 Linux 파일 시스템 권한을 설정하고, 다양한 권한 설정에 따른 보안 영향을 해석합니다.

- 명령줄 툴을 사용하여 파일의 권한 및 소유권을 변경합니다.
- 사용자가 생성한 파일의 기본 권한을 제어하고, 특수 권한의 영향을 설명하고, 특수 권한 및 기본 권한을 사용하여 디렉터리에 생성된 파일의 그룹 소유자를 설정합니다.

5장. SELinux 보안 관리

SELinux를 사용하여 서버의 보안을 보호하고 관리합니다.

- SELinux에서 리소스를 보호하고, 시스템의 현재 SELinux 모드를 변경하고, 시스템의 기본 SELinux 모드를 설정하는 방법을 설명합니다.
- semanage fcontext** 명령을 사용하여 파일 및 디렉터리의 기본 컨텍스트를 결정하고 **restorecon** 명령을 사용하여 SELinux 정책에서 정의한 컨텍스트를 파일 및 디렉터리에 적용하는 SELinux 정책 규칙을 관리합니다.
- setsebool** 명령을 사용하여 SELinux 정책 규칙을 활성화 및 비활성화하고, **semanage boolean -l** 명령을 사용하여 SELinux 부울의 영구 값을 관리하고, **_selinux**로 끝나는 **man** 페이지를 참조하여 SELinux 부울에 대한 유용한 정보를 찾습니다.
- SELinux 로그 분석 툴을 사용하고 **sealert** 명령으로 SELinux 문제를 해결하는 동안 유용한 정보를 표시합니다.

6장. 시스템 성능 튜닝

Red Hat Enterprise Linux 시스템에서 프로세스를 평가 및 제어하고, 튜닝 매개 변수를 설정하며, 프로세스 스케줄링 우선 순위를 조정합니다.

- 명령을 사용하여 프로세스를 강제 종료 및 통신하고 데몬 프로세스의 특성을 정의하고, 사용자 세션 및 프로세스를 중지합니다.
- 부하 평균을 정의하고 리소스를 많이 사용하는 서버 프로세스를 확인합니다.
- 튜닝된 데몬이 관리하는 튜닝 프로파일을 선택하여 시스템 성능을 최적화합니다.
- nice** 및 **renice** 명령을 사용하여 특정 프로세스에 우선 순위를 지정하거나 해제합니다.

7장. 향후 작업 예약

향후 자동으로 실행되는 작업을 예약합니다.

- 사용자의 crontab 파일을 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.
- 시스템 crontab 파일과 디렉터리를 사용하여 반복 스케줄에 따라 실행되도록 명령을 예약합니다.
- systemd 타이머를 활성화 및 비활성화하고 임시 파일을 관리하는 타이머를 구성합니다.

8장. 소프트웨어 패키지 설치 및 업데이트

Red Hat 및 DNF 패키지 리포지토리에서 소프트웨어 패키지를 다운로드하고 설치, 업데이트, 관리합니다.

- Red Hat Subscription Management를 사용하여 Red Hat 계정에 시스템을 등록하고 소프트웨어 업데이트 및 지원 서비스 자격을 할당합니다.
- dnf** 명령을 사용하여 소프트웨어 패키지를 찾고 설치 및 업데이트합니다.

- 서버의 Red Hat 또는 타사 DNF 리포지토리 사용을 활성화 및 비활성화합니다.

9장. 기본 스토리지 관리

명령줄에서 스토리지 장치, 파티션, 파일 시스템 및 스왑 파일을 생성하고 관리합니다.

- 파일 시스템 계층 구조에서 파일 시스템을 추가 및 제거하여 파일 시스템 내용에 액세스합니다.
- 스토리지 파티션을 생성하고, 파일 시스템으로 포맷한 다음, 사용하기 위해 마운트합니다.
- 스왑 공간을 생성하고 관리하여 실제 메모리를 확보합니다.

10장. 스토리지 스택 관리

명령줄에서 파일 시스템 또는 스왑 공간이 포함된 논리 볼륨을 생성 및 관리합니다.

- 논리 볼륨 관리자 구성 요소 및 개념을 설명하고, LVM 스토리지를 구현하고, LVM 구성 요소 정보를 표시합니다.

11장. 제어 서비스 및 부팅 프로세스

`systemd` 서비스를 사용하여 네트워크 서비스, 시스템 데몬 및 부팅 프로세스를 제어 및 모니터링합니다.

- `systemd` 서비스 및 소켓 유닛에서 시작된 시스템 데몬과 네트워크 서비스를 표시합니다.
- `systemctl`을 사용하여 시스템 데몬 및 네트워크 서비스를 제어합니다.
- Red Hat Enterprise Linux 부팅 프로세스를 설명하고, 부팅 시 기본 타겟을 설정하고, 시스템을 기본값이 아닌 타겟으로 부팅합니다.
- 현재 루트 암호가 분실된 경우 시스템에 로그인하여 루트 암호를 변경합니다.
- 부팅 프로세스를 중단하는 파일 시스템 구성 또는 손상 문제를 수동으로 복구합니다.

12장. 로그 분석 및 저장

문제 해결을 위해 시스템 이벤트 로그를 찾아 정확히 해석합니다.

- 이벤트를 기록할 Red Hat Enterprise Linux 기본 로깅 아키텍처에 대해 설명합니다.
- 문제를 해결하거나 시스템 상태를 검토하기 위해 관련 `syslog` 파일의 이벤트를 해석합니다.
- 문제를 해결하거나 시스템 상태를 검토하기 위해 시스템 저널의 로그 항목을 찾아 해석합니다.
- 서버가 재부팅될 때 이벤트 레코드를 보존하도록 시스템 저널을 구성합니다.
- NTP(Network Time Protocol)를 사용하여 정확한 시간 동기화를 유지하고 시스템 저널 및 로그에서 기록하는 이벤트의 타임스탬프가 정확하도록 시간대를 구성합니다.

13장. 네트워킹 관리

Red Hat EnterpriseLinux 서버의 네트워크 인터페이스 및 설정 구성

- 명령줄 유틸리티를 사용하여 현재 네트워크 구성을 테스트 및 검사합니다.
- `nmcli` 명령을 사용하여 네트워크 설정 및 장치를 관리합니다.
- 구성 파일을 편집하여 네트워크 구성을 수정합니다.
- 서버의 정적 호스트 이름과 이름 확인을 구성하고 결과를 테스트합니다.

14장. 네트워크 연결 스토리지 액세스

NFS 프로토콜을 사용하여 네트워크 연결 스토리지에 액세스합니다.

- NFS 내보내기 정보를 식별하고, 마운트 지점으로 사용할 디렉터리를 생성하고, `mount` 명령을 사용하거나 `/etc/fstab` 파일을 구성하여 NFS 내보내기를 마운트하고, `umount` 명령을 사용하여 NFS 내보내기 를 마운트 해제합니다.
- 자동 마운터 사용의 이점을 설명하고 직접 및 간접 맵을 사용하여 NFS 내보내기를 자동으로 마운트합니다.

15장. 네트워크 보안 관리

시스템 방화벽 및 SELinux 규칙을 사용하여 서비스에 대한 네트워크 연결을 제어합니다.

- `firewalld` 규칙을 사용하여 시스템 서비스에 대한 네트워크 연결을 수락하거나 거부합니다.
- 네트워크 포트에 바인딩할 서비스에 맞는 올바른 SELinux 유형이 있는지 확인합니다.

16장. 컨테이너 실행

단일 Red Hat Enterprise Linux Server에서 간단한 경량 서비스를 컨테이너 형태로 가져와 실행하고 관리합니다.

- 컨테이너의 개념과 컨테이너를 빌드, 저장, 실행하는 데 필요한 핵심 기술을 설명합니다.
- 레지스트리를 사용하여 이미지를 저장 및 검색하고, 컨테이너를 배포 및 쿼리하고 컨테이너에 액세스하는 컨테이너 관리 툴을 설명합니다.
- 컨테이너 호스트에서 스토리지를 공유하여 컨테이너 데이터를 위한 영구저장장치를 제공하고 컨테이너 네트워크를 구성합니다.
- 컨테이너를 `systemd` 서비스로 구성하고 부팅 시 시작되도록 컨테이너 서비스를 구성합니다.

▶ 랩

부팅 문제 해결 및 서버 유지 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랩을 시도합니다. 채점 스크립트를 사용하여 각 랩을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 부팅 문제를 해결하고 복구하며 시스템 기본 타겟을 업데이트합니다. 또한 작업이 반복되는 일정에 따라 실행되도록 일반 사용자로 예약합니다.

결과

- 문제를 진단하고 긴급 모드에서 시스템을 복구합니다.
- 기본 대상을 **graphical.target**에서 **multi-user.target**으로 변경합니다.
- 일반 사용자가 실행하도록 반복 실행 작업을 예약합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview1
```

사양

- workstation**에서 **/tmp/rhcsa-break1** 스크립트를 실행합니다. 이 스크립트로 인해 **serverb**의 부팅 프로세스에 문제가 발생하고 시스템이 재부팅됩니다. 부팅 문제의 원인을 해결하고 복구합니다. 암호를 입력하라는 메시지가 표시되면 **root** 사용자의 암호로 **redhat**을 사용합니다.
- workstation**에서 **/tmp/rhcsa-break2** 스크립트를 실행합니다. 이 스크립트로 인해 **serverb** 시스템에서 기본 타겟이 **multi-user** 타겟에서 **graphical** 타겟으로 전환된 다음 시스템이 재부팅됩니다. **serverb**에서 **multi-user** 타겟을 사용하도록 기본 타겟을 재설정합니다. 기본 대상 설정은 수동 개입 없이 재부팅 후에도 유지되어야 합니다. **student** 사용자로 **sudo** 명령을 사용하여 권한 있는 명령을 수행합니다. 필요한 경우 **student**를 암호로 사용합니다.
- serverb**에서 토요일과 일요일을 제외한 모든 요일의 오후 7시~오후 9시 사이에 **/home/student/backup-home.sh** 스크립트를 매시간 실행하는 반복 작업을 **student** 사용자로 예약합니다. <http://materials.example.com/labs/backup-home.sh>에서 백업 스크립트를 다운로드합니다. **backup-home.sh** 스크립트는 **serverb**의 **/home/student** 디렉터리를 **servera**의 **/home/student/serverb-backup** 디렉터리에 백업합니다. **backup-home.sh** 스크립트를 사용하여 반복 실행 작업을 **student** 사용자로 예약합니다. 명령을 실행 파일로 실행합니다.

- **serverb** 시스템을 재부팅하고 부팅이 완료될 때까지 기다린 후 평가합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-comprevew1
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-comprevew1
```

이것으로 섹션을 완료합니다.

▶ 솔루션

부팅 문제 해결 및 서버 유지 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랙을 시도합니다. 채점 스크립트를 사용하여 각 랙을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 부팅 문제를 해결하고 복구하며 시스템 기본 타겟을 업데이트합니다. 또한 작업이 반복되는 일정에 따라 실행되도록 일반 사용자로 예약합니다.

결과

- 문제를 진단하고 긴급 모드에서 시스템을 복구합니다.
- 기본 대상을 **graphical.target**에서 **multi-user.target**으로 변경합니다.
- 일반 사용자가 실행하도록 반복 실행 작업을 예약합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview1
```

1. **workstation**에서 **/tmp/rhcsa-break1** 스크립트를 실행합니다.

```
[student@workstation ~]$ sh /tmp/rhcsa-break1
```

2. **serverb** 시스템이 부팅되면 콘솔에 액세스하여 부팅 프로세스가 초기에 중단되었는지 확인합니다. 이 동작의 원인이 무엇인지 생각합니다.
 - 2.1. 강의실 환경에 해당하는 **serverb** 콘솔의 아이콘을 찾습니다. 콘솔을 열고 오류를 검사합니다. 오류가 표시되는데 몇 초가 걸릴 수 있습니다.
 - 2.2. **Ctrl+Alt+Del**을 눌러 **serverb** 시스템을 재부팅합니다. 부트 로더 메뉴가 나타나면 **Enter** 키를 제외한 임의의 키를 눌러 카운트다운을 중단합니다.
 - 2.3. 메모리의 기본 부트 로더 항목을 편집하여 긴급 모드로 로그인합니다. **e**를 눌러 현재 항목을 편집합니다.
 - 2.4. 커서 키를 사용하여 **linux**로 시작하는 행으로 이동합니다. **systemd.unit=emergency.target**을 추가합니다.

2.5. **Ctrl+x**를 눌러 수정된 구성으로 부팅합니다.

2.6. 긴급 모드로 로그인합니다. **redhat**을 **root** 사용자의 암호로 사용합니다.

```
Give root password for maintenance
(or press Control-D to continue): redhat
[root@serverb ~]#
```

3. 읽기 및 쓰기 기능을 사용하여 / 파일 시스템을 다시 마운트합니다. **mount -a** 명령을 사용하여 기타 모든 파일 시스템을 마운트합니다.

3.1. 읽기 및 쓰기 기능을 사용하여 / 파일 시스템을 다시 마운트하여 파일 시스템을 편집합니다.

```
[root@serverb ~]# mount -o remount,rw /
```

3.2. 나머지 모든 파일 시스템을 마운트해 봅니다. 파일 시스템 중 하나가 마운트되지 않습니다.

```
[root@serverb ~]# mount -a
...output omitted...
mount: /FakeMount: can't find UUID=fake.
```

3.3. **/etc/fstab** 파일을 편집하여 문제를 해결합니다. 잘못된 줄을 제거하거나 주석 처리합니다.

```
[root@serverb ~]# vi /etc/fstab
...output omitted...
#UUID=fake      /FakeMount  xfs    defaults    0 0
```

3.4. 시스템의 **systemd** 데몬을 업데이트하여 새 **/etc/fstab** 파일 구성을 등록합니다.

```
[root@serverb ~]# systemctl daemon-reload
[ 206.828912] systemd[1]: Reloading.
```

3.5. 모든 항목의 마운트를 시도하여 **/etc/fstab**이 올바른지 확인합니다.

```
[root@serverb ~]# mount -a
```

3.6. **serverb**를 재부팅하고 부팅이 완료될 때까지 기다립니다. 이제 시스템이 오류 없이 부팅됩니다.

```
[root@serverb ~]# systemctl reboot
```

4. **workstation**에서 **/tmp/rhcsa-break2** 스크립트를 실행합니다. **serverb** 시스템이 재부팅될 때까지 기다린 후 진행합니다.

```
[student@workstation ~]$ sh /tmp/rhcsa-break2
```

5. **serverb**에서 **multi-user** 타겟을 현재 및 기본 타겟으로 설정합니다.

5.1. **student** 사용자로 **serverb**에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

5.2. 기본 타겟을 결정합니다.

```
[student@serverb ~]$ systemctl get-default
graphical.target
```

5.3. 전환 **multi-user** 목표.

```
[student@serverb ~]$ sudo systemctl isolate multi-user.target
[sudo] password for student: student
```

5.4. **multi-user** 타겟을 기본 타겟으로 설정합니다.

```
[student@serverb ~]$ sudo systemctl set-default multi-user.target
Removed /etc/systemd/system/default.target.
Created symlink /etc/systemd/system/default.target → /usr/lib/systemd/system/
multi-user.target.
```

5.5. **serverb**를 재부팅하여 **multi-user** 타겟이 기본 타겟으로 설정되어 있는지 확인합니다.

```
[student@serverb ~]$ sudo systemctl reboot
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

5.6. 시스템이 재부팅되면 **student** 사용자로 **serverb**에 대한 SSH 세션을 엽니다. **multi-user** 대상이 기본 대상으로 설정되어 있는지 확인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ systemctl get-default
multi-user.target
```

6. **serverb**에서 토요일과 일요일을 제외한 모든 요일의 오후 7시~오후 9시 사이에 **/home/student/backup-home.sh** 스크립트를 매시간 실행하는 반복 작업을 **student** 사용자로 예약합니다. **backup-home.sh** 스크립트를 사용하여 반복 실행 작업을 예약합니다. **http://materials.example.com/labs/backup-home.sh**에서 백업 스크립트를 다운로드합니다. 명령을 실행 파일로 실행합니다.

6.1. **serverb**에서 **http://materials.example.com/labs/backup-home.sh**에 있는 백업 스크립트를 다운로드합니다. **chmod**을 사용하여 백업 스크립트를 실행 파일로 만듭니다.

```
[student@serverb ~]$ wget http://materials.example.com/labs/backup-home.sh
...output omitted...
[student@serverb ~]$ chmod +x backup-home.sh
```

6.2. 기본 텍스트 편집기에서 crontab 파일을 엽니다.

```
[student@serverb ~]$ crontab -e
```

6.3. 파일을 편집하고 다음 행을 추가합니다.

```
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

변경 사항을 저장하고 편집기를 종료합니다.

6.4. **crontab -l** 명령을 사용하여 예약된 반복 실행 작업을 나열합니다.

```
[student@serverb ~]$ crontab -l
0 19-21 * * Mon-Fri /home/student/backup-home.sh
```

7. **serverb**를 재부팅하고 부팅이 완료될 때까지 기다린 후 평가합니다.

```
[student@serverb ~]$ sudo systemctl reboot
[sudo] password for student: student
Connection to serverb closed by remote host.
Connection to serverb closed.
[student@workstation ~]$
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-comprevew1
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-comprevew1
```

이것으로 섹션을 완료합니다.

▶ 랩

파일 시스템 및 스토리지 구성 및 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랙을 시도합니다. 채점 스크립트를 사용하여 각 랙을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 논리 볼륨을 생성하고, 네트워크 파일 시스템을 마운트하고, 부팅 시 자동으로 활성화되는 스왑 파티션을 생성합니다. 또한 임시 파일을 저장하도록 디렉터리를 구성합니다.

결과

- 논리 볼륨을 만듭니다.
- 네트워크 파일 시스템을 마운트합니다.
- 부팅 시 자동으로 활성화되는 스왑 파티션을 만듭니다.
- 임시 파일을 저장하도록 디렉터리를 구성합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview2
```

사양

- serverb**에서 새 2GiB **extra_storage** 볼륨 그룹에 새 1GiB **vol_home** 논리 볼륨을 구성합니다. 파티셔닝되지 않은 **/dev/vdb** 디스크를 사용하여 파티션을 생성합니다.
- 논리 볼륨 **vol_home**은 XFS 파일 시스템 유형으로 포맷하고 **/user-homes**에 영구적으로 마운트해야 합니다.
- serverb**에서 **servera**가 **/local-share** 디렉터리에서 내보내는 **/share** 네트워크 파일 시스템을 영구적으로 마운트합니다. **servera** 시스템에서 **servera.lab.example.com:/share** 경로를 내보냅니다.
- serverb**에서 **/dev/vdc** 디스크에 512MiB 스왑 파티션을 생성합니다. 스왑 파티션을 영구적으로 마운트합니다.
- production** 사용자 그룹을 생성합니다. **production** 그룹을 보조 그룹으로 사용하여 **production1**, **production2**, **production3**, **production4** 사용자를 생성합니다.

- **serverb**에서 임시 파일을 저장하도록 **/run/volatile** 디렉터리를 구성합니다. 이 디렉터리의 파일에 30초 이상 액세스하지 않으면 시스템에서 자동으로 삭제합니다. 디렉터리에 대한 8진수 권한으로 **0700**을 설정합니다. **/etc/tmpfiles.d/volatile.conf** 파일을 사용하여 **/run/volatile** 디렉터리에서 시간 기반 파일 삭제를 구성합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-comprevew2
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-comprevew2
```

이것으로 섹션을 완료합니다.

▶ 솔루션

파일 시스템 및 스토리지 구성 및 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랙을 시도합니다. 채점 스크립트를 사용하여 각 랙을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 논리 볼륨을 생성하고, 네트워크 파일 시스템을 마운트하고, 부팅 시 자동으로 활성화되는 스왑 파티션을 생성합니다. 또한 임시 파일을 저장하도록 디렉터리를 구성합니다.

결과

- 논리 볼륨을 만듭니다.
- 네트워크 파일 시스템을 마운트합니다.
- 부팅 시 자동으로 활성화되는 스왑 파티션을 만듭니다.
- 임시 파일을 저장하도록 디렉터리를 구성합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview2
```

1. **serverb**에서 새 2GiB **extra_storage** 볼륨 그룹에 새 1GiB **vol_home** 논리 볼륨을 구성합니다. 파티셔닝되지 않은 **/dev/vdb** 디스크를 사용하여 파티션을 생성합니다.

- 1.1. **serverb**에 **student** 사용자로 로그인한 후 **root** 사용자로 전환합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 1.2. **/dev/vdb** 디스크에 2MiB 파티션을 생성합니다.

```
[root@serverb ~]# parted /dev/vdb mklabel msdos
...output omitted...
[root@serverb ~]# parted /dev/vdb mkpart primary 1MiB 2GiB
...output omitted...
[root@serverb ~]# parted /dev/vdb set 1 lvm on
...output omitted...
```

1.3. `/dev/vdb1` 블록 장치를 물리적 볼륨으로 정의합니다.

```
[root@serverb ~]# pvcreate /dev/vdb1
...output omitted...
```

1.4. `/dev/vdb1` 파티션이 있는 `extra_storage` 볼륨 그룹을 생성합니다.

```
[root@serverb ~]# vgcreate extra_storage /dev/vdb1
...output omitted...
```

1.5. 1GiB `vol_home` 논리 볼륨을 생성합니다.

```
[root@serverb ~]# lvcreate -L 1GiB -n vol_home extra_storage
...output omitted...
```

2. 논리 볼륨 `vol_home`은 XFS 파일 시스템 유형으로 포맷하고 `/user-homes`에 영구적으로 마운트해야 합니다.

2.1. `/user-homes` 디렉터리를 생성합니다.

```
[root@serverb ~]# mkdir /user-homes
```

2.2. `/dev/extra_storage/vol_home` 파티션을 XFS 파일 시스템 유형으로 포맷합니다.

```
[root@serverb ~]# mkfs -t xfs /dev/extra_storage/vol_home
...output omitted...
```

2.3. `/dev/extra_storage/vol_home` 파티션을 `/user-homes` 디렉터리에 영구적으로 마운트합니다. `/etc/fstab` 파일 항목에 파티션의 UUID를 사용합니다.

```
[root@serverb ~]# lsblk -o UUID /dev/extra_storage/vol_home
UUID
988cf149-0667-4733-abca-f80c6ec50ab6
[root@serverb ~]# echo "UUID=988cf149-0667-4733-abca-f80c6ec50ab6 /user-homes xfs defaults 0 0" \
>> /etc/fstab
[root@serverb ~]# mount /user-homes
```

3. `serverb`에서 `servera`가 `/local-share` 디렉터리에서 내보내는 `/share` 네트워크 파일 시스템을 영구적으로 마운트합니다. `servera` 시스템에서 `servera.lab.example.com:/share` 경로를 내보냅니다.

3.1. `/local-share` 디렉터리를 생성합니다.

```
[root@serverb ~]# mkdir /local-share
```

- 3.2. `servera.lab.example.com:/share` 네트워크 파일 시스템에 영구적으로 마운트되도록 `/etc/fstab` 파일에 적절한 항목을 추가합니다.

```
[root@serverb ~]# echo "servera.lab.example.com:/share /local-share \
nfs rw,sync 0 0" >> /etc/fstab
```

- 3.3. `/local-share` 디렉터리에 네트워크 파일 시스템을 마운트합니다.

```
[root@serverb ~]# mount /local-share
```

4. `serverb`에서 `/dev/vdc` 디스크에 512MiB 스왑 파티션을 생성합니다. 스왑 파티션을 활성화하고 영구적으로 마운트합니다.

- 4.1. `/dev/vdc` 디스크에 512MiB 파티션을 생성합니다.

```
[root@serverb ~]# parted /dev/vdc mklabel msdos
...output omitted...
[root@serverb ~]# parted /dev/vdc mkpart primary linux-swap 1MiB 513MiB
...output omitted...
```

- 4.2. `/dev/vdc1` 파티션에 스왑 공간을 생성합니다.

```
[root@serverb ~]# mkswap /dev/vdc1
...output omitted...
```

- 4.3. `/etc/fstab` 파일에 항목을 생성하여 스왑 공간을 영구적으로 마운트합니다. 파티션의 UUID를 사용하여 `/etc/fstab` 파일 항목을 생성합니다. 스왑 공간을 활성화합니다.

```
[root@serverb ~]# lsblk -o UUID /dev/vdc1
UUID
cc18ccb6-bd29-48a5-8554-546bf3471b69
[root@serverb ~]# echo "UUID=cc18...1b69 swap swap defaults 0 0" >> /etc/fstab
[root@serverb ~]# swapon -a
```

5. `production` 사용자 그룹을 생성합니다. 그런 다음 `production` 그룹을 보조 그룹으로 사용하여 `production1, production2, production3, production4` 사용자를 생성합니다.

```
[root@serverb ~]# groupadd production
[root@serverb ~]# for i in 1 2 3 4; do useradd -G production production$i; done
```

6. `serverb`에서 임시 파일을 저장하도록 `/run/volatile` 디렉터리를 구성합니다. 이 디렉터리의 파일이 30초 이상 액세스하지 않으면 시스템에서 자동으로 삭제합니다. 디렉터리에 대한 8진수 권한으로 `0700`을 설정합니다. `/etc/tmpfiles.d/volatile.conf` 파일을 사용하여 `/run/volatile` 디렉터리에서 시간 기반 파일 삭제를 구성합니다.

- 6.1. 다음 내용으로 `/etc/tmpfiles.d/volatile.conf` 파일을 생성합니다.

```
d /run/volatile 0700 root root 30s
```

- 6.2. `/run/volatile` 디렉터리가 없는 경우 `systemd-tmpfiles --create` 명령을 사용하여 만듭니다.

```
[root@serverb ~]# systemd-tmpfiles --create /etc/tmpfiles.d/volatile.conf
```

- 6.3. `workstation` 시스템에 `student` 사용자로 돌아갑니다.

```
[root@serverb ~]# exit
logout
[student@serverb ~]$ exit
logout
Connection to serverb closed.
```

평가

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-comprevew2
```

완료

`workstation` 시스템에서 `student` 사용자 홈 디렉터리로 변경하고 `lab` 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-comprevew2
```

이것으로 섹션을 완료합니다.

▶ 랩

서버 보안 구성 및 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랩을 시도합니다. 채점 스크립트를 사용하여 각 랩을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 SSH 키 기반 인증 구성, 방화벽 설정 변경, SELinux 모드 및 SELinux 부울 조정, SELinux 문제 해결을 수행합니다.

결과

- SSH 키 기반 인증을 구성합니다.
- 방화벽 설정을 구성합니다.
- SELinux 모드 및 SELinux 부울을 조정합니다.
- SELinux 문제를 해결합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview3
```

사양

- serverb**에서 **student** 사용자에 대한 SSH 키 쌍을 생성합니다. 개인 키를 암호로 보호하면 안 됩니다.
- serverb** 시스템에서 생성한 SSH 키 쌍을 사용하여 로그인 인증을 수락하도록 **servera**에 **student** 사용자를 구성합니다. **serverb**의 **student** 사용자는 암호를 입력하지 않고 SSH를 통해 **servera**에 로그인할 수 있어야 합니다.
- servera**에서 **/user-homes/production5** 디렉터리 권한을 확인합니다. 그런 다음 기본적으로 **permissive** 모드로 실행되도록 SELinux를 구성합니다.
- serverb**에서 **/localhome** 디렉터리가 없음을 확인합니다. 그런 다음 **/user-homes/production5** 네트워크 파일 시스템을 마운트하도록 **production5** 사용자의 홈 디렉터리를 구성합니다. **servera.lab.example.com** 시스템은 **servera.lab.example.com:/user-homes/production5** NFS 공유로 파일 시스템을 내보냅니다. **autofs** 서비스를 사용하여 네트워크 공유를 마

운트합니다. **autofs** 서비스가 **servera**에서와 동일한 권한으로 **/localhome/production5** 디렉터리를 생성하는지 확인합니다.

- **serverb**에서 SSH 키를 통해 인증이 완료되면 **production5** 사용자가 NFS로 마운트된 홈 디렉터리를 사용할 수 있도록 해당 SELinux 부울을 조정합니다. 필요한 경우 **redhat**을 **production5** 사용자의 암호로 사용합니다.
- **serverb**에서 방화벽 설정을 조정하여 **servera** 시스템의 모든 연결 요청을 거부합니다. **servera** IPv4 주소(**172.25.250.10**)를 사용하여 방화벽 규칙을 구성합니다.
- **serverb**에서 연결을 위해 포트 **30080/TCP**에서 수신 대기하는 실패한 Apache 웹 서비스의 문제를 조사하고 해결합니다. 포트 **30080/TCP**가 들어오는 연결에 대해 열리도록 방화벽 설정을 적절하게 조정합니다.

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-comprevew3
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-comprevew3
```

이것으로 섹션을 완료합니다.

▶ 솔루션

서버 보안 구성 및 관리



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랙을 시도합니다. 채점 스크립트를 사용하여 각 랙을 완료하면서 진행 상황을 평가합니다.

이 검토에서는 SSH 키 기반 인증 구성, 방화벽 설정 변경, SELinux 모드 및 SELinux 부울 조정, SELinux 문제 해결을 수행합니다.

결과

- SSH 키 기반 인증을 구성합니다.
- 방화벽 설정을 구성합니다.
- SELinux 모드 및 SELinux 부울을 조정합니다.
- SELinux 문제를 해결합니다.

시작하기 전에

마지막 장이 끝날 때 **workstation** 및 **server** 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview3
```

- server**에서 **student** 사용자에 대한 SSH 키 쌍을 생성합니다. 개인 키를 암호로 보호하면 안 됩니다.

1. **student** 사용자로 **server**에 로그인합니다.

```
[student@workstation ~]$ ssh student@server
...output omitted...
```

2. **ssh-keygen** 명령을 사용하여 SSH 키 쌍을 생성합니다. 개인 키를 암호로 보호하면 안 됩니다.

```
[student@server ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/student/.ssh/id_rsa): Enter
Created directory '/home/student/.ssh'.
```

```
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
Your identification has been saved in /home/student/.ssh/id_rsa.
Your public key has been saved in /home/student/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:+ijpGqjEQSGBR80RNchiRTHw/URQksVdHjsHqVBXeYI student@serverb.lab.example.com
The key's randomart image is:
+---[RSA 3072]---+
|+BBX+o*+o..=+.. |
|+.0.oooo .oE+o . |
|.+ . . . .+ .o |
|. o . o . o |
| . .S . |
|... . |
|o. . . |
|o . o o |
|..o.... |
+---[SHA256]-----+
```

2. **serverb** 시스템에서 생성한 SSH 키 쌍을 사용하여 로그인 인증을 수락하도록 **servera**에 **student** 사용자를 구성합니다. **serverb**의 **student** 사용자는 암호를 입력하지 않고 SSH를 통해 **servera**에 로그인할 수 있어야 합니다.

- 2.1. 새로 생성된 SSH 키 쌍의 공개 키를 **servera** 시스템의 **student** 사용자에게 보냅니다.

```
[student@serverb ~]$ ssh-copy-id student@servera
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/
id_rsa.pub"
The authenticity of host 'servera (172.25.250.10)' can't be established.
ED25519 key fingerprint is SHA256:shYfoFG0Nnv42pv7j+HG+FISmCAm4Bh5jfjwwSMJbrw.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
student@servera's password: student

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'student@servera'"
and check to make sure that only the key(s) you wanted were added.
```

- 2.2. **student** 사용자가 암호를 입력하지 않고 **serverb**에서 **servera**에 로그인할 수 있는지 확인합니다. 연결을 종료하지 마십시오.

```
[student@serverb ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

3. **servera**에서 **/user-homes/production5** 디렉터리 권한을 확인합니다. 그런 다음 기본적으로 **permissive** 모드로 실행되도록 SELinux를 구성합니다.

- 3.1. **/user-homes/production5** 디렉터리 권한을 확인합니다.

```
[student@servera ~]$ ls -ld /user-homes/production5
drwx----- 2 production5 production5 62 May  6 05:27 /user-homes/production5
```

- 3.2. `/etc/sysconfig/selinux` 파일을 편집하여 SELINUX 매개 변수 값을 `permissive`로 설정합니다.

```
[student@servera ~]$ sudo vi /etc/sysconfig/selinux
...output omitted...
#SELINUX=enforcing
SELINUX=permissive
...output omitted...
```

- 3.3. 시스템을 재부팅합니다.

```
[student@servera ~]$ sudo systemctl reboot
Connection to servera closed by remote host.
Connection to servera closed.
[student@serverb ~]$
```

4. `serverb`에서 `/localhome` 디렉터리가 없음을 확인합니다. 그런 다음 `/user-homes/production5` 네트워크 파일 시스템을 마운트하도록 `production5` 사용자의 홈 디렉터리를 구성합니다. `servera.lab.example.com` 시스템은 `servera.lab.example.com:/user-homes/production5` NFS 공유로 파일 시스템을 내보냅니다. `autofs` 서비스를 사용하여 네트워크 공유를 마운트합니다. `autofs` 서비스가 `servera`에서와 동일한 권한으로 `/localhome/production5` 디렉터리를 생성하는지 확인합니다.

- 4.1. `/localhome` 디렉터리가 없음을 확인합니다.

```
[student@serverb ~]$ ls -ld /localhome
ls: cannot access '/localhome': No such file or directory
```

- 4.2. `serverb`에서 `root` 사용자로 전환합니다.

```
[student@serverb ~]$ sudo -i
[sudo] password for student: student
[root@serverb ~]#
```

- 4.3. `autofs` 패키지를 설치합니다.

```
[root@serverb ~]# dnf install autofs
...output omitted...
Is this ok [y/N]: y
...output omitted...
Installed:
  autofs-1:5.1.7-27.el9.x86_64      libssss_autofs-2.6.2-2.el9.x86_64

Complete!
```

- 4.4. 다음 내용으로 `/etc/auto.master.d/production5.autofs` 맵 파일을 만듭니다.

```
/- /etc/auto.production5
```

- 4.5. **production5** 사용자의 홈 디렉터리를 결정합니다.

```
[root@serverb ~]# getent passwd production5
production5:x:5001:5001::/localhome/production5:/bin/bash
```

- 4.6. 다음 내용으로 **/etc/auto.production5** 파일을 생성합니다.

```
/localhome/production5 -rw servera.lab.example.com:/user-homes/production5
```

- 4.7. **autofs** 서비스를 다시 시작합니다.

```
[root@serverb ~]# systemctl restart autofs
```

- 4.8. **autofs** 서비스가 **servera**의 **/user-homes/production5** 디렉터리와 동일한 권한으로 **serverb**에 **/localhome/production5** 디렉터리를 생성하는지 확인합니다.

```
[root@serverb ~]# ls -ld /localhome/production5
drwx----- 2 production5 production5 62 May  6 05:52 /localhome/production5
```

- 5.** **serverb**에서 SSH 키를 통해 인증이 완료되면 **production5** 사용자가 NFS로 마운트된 홈 디렉터리를 사용할 수 있도록 해당 SELinux 부울을 조정합니다. 필요한 경우 **redhat** 을 **production5** 사용자의 암호로 사용합니다.

- 5.1. 새 터미널 창을 열고 **servera**에서 **production5** 사용자가 SSH 키 기반 인증을 사용하여 **serverb**에 로그인할 수 있는지 확인합니다. SELinux 부울로 인해 사용자가 로그인할 수 없습니다. **workstation**에서 새 터미널을 열고 **servera**에 **student** 사용자로 로그인합니다.

```
[student@workstation ~]$ ssh student@servera
...output omitted...
[student@servera ~]$
```

- 5.2. **production5** 사용자로 전환합니다. 암호를 입력하라는 메시지가 표시되면 **production5** 사용자의 암호로 **redhat**을 사용합니다.

```
[student@servera ~]$ su - production5
Password: redhat
[production5@servera ~]$
```

- 5.3. SSH 키 쌍을 생성합니다.

```
[production5@servera ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/production5/.ssh/id_rsa): Enter
Created directory '/home/production5/.ssh'.
Enter passphrase (empty for no passphrase): Enter
Enter same passphrase again: Enter
```

```
Your identification has been saved in /home/production5/.ssh/id_rsa.
Your public key has been saved in /home/production5/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:AbUcIBXneyiGIhr4wS1xzs3WqDvbTP+eZuSRn9HQ/cw
    production5@servera.lab.example.com
The key's randomart image is:
+---[RSA 3072]----+
|       ..=++      |
|       . = o      |
|     . . = . . . |
| .. * + o + . . .|
|+= = B S .. o o .|
|.+ + + .+. . E|
|.. . . . o o o   |
|     .= . +.o     |
|     ooo .=+     |
+---[SHA256]-----+
```

- 5.4. SSH 키 쌍의 공개 키를 **serverb** 시스템의 **production5** 사용자에게 전송합니다. 암호를 입력하라는 메시지가 표시되면 **production5** 사용자의 암호로 **redhat**을 사용합니다.

```
[production5@servera ~]$ ssh-copy-id production5@serverb
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/
production5/.ssh/id_rsa.pub"
The authenticity of host 'serverb (172.25.250.11)' can't be established.
ECDSA key fingerprint is SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted
now it is to install the new keys
production5@serverb's password: redhat

Number of key(s) added: 1

Now try logging in to the machine, with: "ssh 'production5@serverb'"
and check to make sure that only the key(s) you wanted were added.
```

- 5.5. 암호 기반 인증 대신 SSH 공개 키 기반 인증을 사용하여 **serverb**에 **production5** 사용자로 로그인합니다. 이 명령은 실패합니다.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-
mic,password).
```

- 5.6. **serverb**에 **root** 사용자로 연결된 터미널에서 **use_nfs_home_dirs** SELinux 부울을 **true**로 설정합니다.

```
[root@serverb ~]# setsebool -P use_nfs_home_dirs true
```

- 5.7. **servera**에 **production5** 사용자로 연결된 터미널로 돌아가서 암호 기반 인증 대신 SSH 공개 키 기반 인증을 사용하여 **serverb**에 **production5** 사용자로 로그인합니다. 이 명령은 성공합니다.

```
[production5@servera ~]$ ssh -o pubkeyauthentication=yes \
-o passwordauthentication=no production5@serverb
...output omitted...
[production5@serverb ~]$
```

- 5.8. **production5** 사용자로 **serverb**에 연결된 터미널을 종료하고 닫습니다. **root** 사용자로 **serverb**에 연결된 터미널을 계속 열어둡니다.
6. **serverb**에서 방화벽 설정을 조정하여 **servera** 시스템에서 발생하는 모든 연결 요청을 거부합니다. **servera** IPv4 주소(**172.25.250.10**)를 사용하여 방화벽 규칙을 구성합니다.

- 6.1. **servera**의 IPv4 주소를 **block** 영역에 추가합니다.

```
[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \
--zone=block --permanent
success
```

- 6.2. 방화벽 설정 변경 사항을 다시 로드합니다.
7. **serverb**에서 연결을 위해 포트 **30080/TCP**에서 수신 대기하는 실패한 Apache 웹 서비스의 문제를 조사하고 해결합니다. 포트 **30080/TCP**가 들어오는 연결에 대해 열리도록 방화벽 설정을 적절하게 조정합니다.

- 7.1. **httpd** 서비스를 다시 시작합니다. 이 명령에서 서비스를 다시 시작하지 못합니다.

```
[root@serverb ~]# systemctl restart httpd.service
Job for httpd.service failed because the control process exited with error code.
See "systemctl status httpd.service" and "journalctl -xeu httpd.service" for
details.
```

- 7.2. **httpd** 서비스가 실패하는 이유를 조사합니다. 시작 시 **httpd** 데몬이 포트 **30080/TCP**에 바인딩되지 않았음을 나타내는 권한 오류가 표시됩니다. SELinux 정책을 사용하여 애플리케이션 이 비표준 포트에 바인딩되는 것을 방지할 수 있습니다. 명령을 종료하려면 **q** 를 누릅니다.

```
[root@serverb ~]# systemctl status httpd.service
× httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor
preset: disabled)
     Active: failed (Result: exit-code) since Mon 2022-05-02 13:20:46 EDT; 29s ago
       Docs: man:httpd.service(8)
    Process: 2322 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited,
   status=1/FAILURE)
      Main PID: 2322 (code=exited, status=1/FAILURE)
        Status: "Reading configuration..."
```

```
CPU: 30ms

May 02 13:20:46 serverb.lab.example.com systemd[1]: Starting The Apache HTTP
Server...
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied:
AH00072: make_sock: could not bind to address [::]:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: (13)Permission denied:
AH00072: make_sock: could not bind to address 0.0.0.0:30080
May 02 13:20:46 serverb.lab.example.com httpd[2322]: no listening sockets
available, shutting down
...output omitted...
```

7.3. SELinux 정책으로 인해 **httpd** 서비스가 **30080/TCP** 포트에 바인딩되지 않는지 확인합니다. 로그 메시지를 통해 **30080/TCP** 포트에 적절한 **http_port_t** SELinux 컨텍스트가 없으며 이로 인해 SELinux에서 **httpd** 서비스가 해당 포트에 바인딩되지 않도록 한다는 사실을 알 수 있습니다. 또한 로그 메시지에 **semanage port** 명령의 구문이 생성되어 이를 통해 문제를 해결할 수 있습니다.

```
[root@serverb ~]# sealert -a /var/log/audit/audit.log
...output omitted...
SELinux is preventing /usr/sbin/httpd from name_bind access on the tcp_socket port
30080.

***** Plugin bind_ports (92.2 confidence) suggests *****

If you want to allow /usr/sbin/httpd to bind to network port 30080
Then you need to modify the port type.
Do
# semanage port -a -t PORT_TYPE -p tcp 30080
  where PORT_TYPE is one of the following: http_cache_port_t, http_port_t,
  jboss_management_port_t, jboss.messaging_port_t, ntop_port_t, puppet_port_t.
...output omitted...
```

7.4. **httpd** 서비스가 **30080/TCP** 포트에 바인딩되도록 해당 포트에 적절한 SELinux 컨텍스트를 설정합니다.

```
[root@serverb ~]# semanage port -a -t http_port_t -p tcp 30080
```

7.5. **httpd** 서비스를 다시 시작합니다. 이 명령이 서비스를 다시 시작해야 합니다.

```
[root@serverb ~]# systemctl restart httpd
```

7.6. **30080/TCP** 포트를 기본 **public** 영역에 추가합니다.

```
[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent
success
[root@serverb ~]# firewall-cmd --reload
success
```

7.7. **workstation** 시스템에 **student** 사용자로 돌아갑니다.

```
[root@serverb ~]# exit  
logout  
[student@serverb ~]$ exit  
logout  
Connection to serverb closed.
```

평가

workstation 시스템에서 **student** 사용자로 **lab** 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-compreview3
```

완료

workstation 시스템에서 **student** 사용자 홈 디렉터리로 변경하고 **lab** 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-compreview3
```

이것으로 섹션을 완료합니다.

▶ 랩

컨테이너 실행



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랩을 시도합니다. 채점 스크립트를 사용하여 각 랩을 완료하면서 진행 상황을 평가합니다.

결과

- 분리된 rootless 컨테이너를 만듭니다.
- 포트 매핑 및 영구저장장치를 구성합니다.
- 컨테이너에서 관리할 수 있도록 `systemctl` 명령을 사용하여 `systemd`를 구성합니다.

시작하기 전에

마지막 장이 끝날 때 `workstation` 및 `server` 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-compreview4
```

사양

- `serverb`에서 `redhat`을 암호로 사용하여 `podmgr` 사용자를 구성하고, `podmgr` 사용자가 이 종합 검토를 위해 컨테이너를 관리할 수 있도록 적절한 툴을 설정합니다. `registry.lab.example.com`을 원격 레지스트리로 구성합니다. `admin`을 사용자로, `redhat321`을 암호를 사용하여 인증합니다. `/tmp/review4/registries.conf` 파일을 사용하여 레지스트리를 구성할 수 있습니다.
- `/tmp/review4/container-dev` 디렉터리에는 이 종합 검토의 컨테이너에 대한 개발 파일이 포함된 두 개의 디렉터리가 있습니다. `/tmp/review4/container-dev` 디렉터리 아래의 두 디렉터리를 `podmgr` 홈 디렉터리에 복사합니다. `/home/podmgr/storage/database` 하위 디렉터리를 컨테이너의 영구저장장치로 사용할 수 있도록 구성합니다.
- `production` DNS 사용 컨테이너 네트워크를 생성합니다. `10.81.0.0/16` 서브넷 및 `10.81.0.1`을 게이트웨이로 사용합니다. 이 종합 검토에서 생성하는 컨테이너에 이 컨테이너 네트워크를 사용합니다.
- `production` 네트워크에서 태그 번호가 가장 낮은 `registry.lab.example.com/rhel8/mariadb-103` 컨테이너 이미지를 기반으로 분리된 컨테이너 `db-app01`을 생성합니다. `/home/podmgr/storage/database` 디렉터리를 `db-app01` 컨테이너의 `/var/lib/mysql/data` 디렉터리를 위한 영구저장장치로 사용합니다. 로컬 시스템의 13306 포트를 컨테이너의 3306 포트에 매핑합니다. 다음 표에 있는 값을 사용하여 컨테이너화된 데이터베이스를 생성하도록 환경 변수를 설정합니다.

변수	값
MYSQL_USER	developer
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

- db-app01 컨테이너를 관리할 **systemd** 서비스 파일을 생성합니다. 서비스를 시작할 때 **systemd** 데몬이 원본 컨테이너를 유지하도록 **systemd** 서비스를 구성합니다. 컨테이너를 **systemd** 서비스로 시작하고 활성화합니다. 시스템 부팅 시 시작하도록 db-app01 컨테이너를 구성합니다.
- /home/podmgr/db-dev/inventory.sql 스크립트를 db-app01 컨테이너의 /tmp 디렉터리에 복사하고 스크립트를 컨테이너 내부에서 실행합니다. 스크립트를 로컬로 실행한 경우 mysql -u root inventory < /tmp/inventory.sql 명령을 사용합니다.
- /home/podmgr/http-dev 디렉터리의 컨테이너 파일을 사용하여 production 네트워크에서 분리된 컨테이너 http-app01을 생성합니다. 컨테이너 이미지 이름은 9.0 태그가 있는 http-client여야 합니다. 로컬 시스템의 8080 포트를 컨테이너의 8080 포트에 매핑합니다.
- curl 명령을 사용하여 http-app01 컨테이너의 콘텐츠를 쿼리합니다. 명령 출력에 클라이언트의 컨테이너 이름이 표시되고 데이터베이스 상태가 'up'인지 확인합니다.

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-compreview4
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-compreview4
```

이것으로 섹션을 완료합니다.

▶ 솔루션

컨테이너 실행



참고

RHCSA 시험에 응시하려는 경우 다음 방법을 사용하여 종합 검토의 이점을 극대화합니다. 솔루션 버튼을 보거나 교육 과정 내용을 참조하지 않고 각 랙을 시도합니다. 채점 스크립트를 사용하여 각 랙을 완료하면서 진행 상황을 평가합니다.

결과

- 분리된 rootless 컨테이너를 만듭니다.
- 포트 매팅 및 영구저장장치를 구성합니다.
- 컨테이너에서 관리할 수 있도록 `systemctl` 명령을 사용하여 `systemd`를 구성합니다.

시작하기 전에

마지막 장이 끝날 때 `workstation` 및 `server` 시스템을 재설정하지 않은 경우 해당 시스템에서 유지하려는 이전 연습의 모든 작업을 저장한 후 지금 재설정합니다.

`workstation` 시스템에서 `student` 사용자로 `lab` 명령을 사용하여 이 연습에 사용할 시스템을 준비합니다.

이 명령은 환경을 준비하고 필요한 모든 리소스를 사용할 수 있는지 확인합니다.

```
[student@workstation ~]$ lab start rhcsa-comprevew4
```

- `serverb`에서 `redhat`을 암호로 사용하여 `podmgr` 사용자를 구성하고, `podmgr` 사용자가 이 종합 검토를 위해 컨테이너를 관리할 수 있도록 적절한 툴을 설정합니다. `registry.lab.example.com`을 원격 레지스트리로 구성합니다. `admin`을 사용자로, `redhat321`을 암호를 사용하여 인증합니다. `/tmp/review4/registries.conf` 파일을 사용하여 레지스트리를 구성할 수 있습니다.

1. `student` 사용자로 `serverb`에 로그인합니다.

```
[student@workstation ~]$ ssh student@serverb
...output omitted...
[student@serverb ~]$
```

2. `container-tools` 메타 패키지를 설치합니다.

```
[student@serverb ~]$ sudo dnf install container-tools
[sudo] password for student: student
...output omitted...
Is this ok [y/N]: y
...output omitted...
Complete!
```

- 1.3. `podmgr` 사용자를 생성하고 사용자의 암호를 `redhat`으로 설정합니다.

```
[student@serverb ~]$ sudo useradd podmgr
[student@serverb ~]$ sudo passwd podmgr
Changing password for user podmgr.
New password: redhat
BAD PASSWORD: The password is shorter than 8 characters
Retype new password: redhat
passwd: all authentication tokens updated successfully.
```

- 1.4. `student` 사용자 세션을 종료합니다. `podmgr` 사용자로 `serverb` 시스템에 로그인합니다. 메시지가 표시되면 암호로 `redhat`을 사용합니다.

```
[student@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$ ssh podmgr@serverb
...output omitted...
[podmgr@serverb ~]$
```

- 1.5. `~/.config/containers` 디렉터리를 생성합니다.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/containers
```

- 1.6. `/tmp/review4/registries.conf` 파일을 홈 디렉터리의 컨테이너 구성 디렉터리에 복사합니다.

```
[podmgr@serverb ~]$ cp /tmp/review4/registries.conf ~/.config/containers/
```

- 1.7. 레지스트리에 로그인하여 구성을 확인합니다.

```
[podmgr@serverb ~]$ podman login registry.lab.example.com
Username: admin
Password: redhat321
Login Succeeded!
```

2. `/tmp/review4/container-dev` 디렉터리에는 이 종합 검토의 컨테이너에 대한 개발 파일이 포함된 두 개의 디렉터리가 있습니다. `/tmp/review4/container-dev` 디렉터리 아래의 두 디렉터리를 `podmgr` 홈 디렉터리에 복사합니다. `/home/podmgr/storage/database` 하위 디렉터리를 컨테이너의 영구저장장치로 사용할 수 있도록 구성을 합니다.

- 2.1. `/tmp/review4/container-dev` 디렉터리의 콘텐츠를 `podmgr` 홈 디렉터리에 복사합니다.

```
[podmgr@serverb ~]$ cp -r /tmp/review4/container-dev/* .
[podmgr@serverb ~]$ ls -l
total 0
drwxr-xr-x. 2 podmgr podmgr 27 May 10 21:52 db-dev
drwxr-xr-x. 2 podmgr podmgr 44 May 10 21:52 http-dev
```

- 2.2. `podmgr` 홈 디렉터리에 `/home/podmgr/storage/database` 디렉터리를 생성합니다. 컨테이너에서 디렉터리를 영구저장장치로 마운트하도록 디렉터리에 적절한 권한을 설정합니다.

```
[podmgr@serverb ~]$ mkdir -p storage/database
[podmgr@serverb ~]$ chmod 0777 storage/database
[podmgr@serverb ~]$ ls -l storage/
total 0
drwxrwxrwx. 2 podmgr podmgr 6 May 10 21:55 database
```

3. `production` DNS 사용 컨테이너 네트워크를 생성합니다. `10.81.0.0/16` 서브넷 및 `10.81.0.1`을 게이트웨이로 사용합니다. 이 종합 검토에서 생성하는 컨테이너에 이 컨테이너 네트워크를 사용합니다.

- 3.1. `production` DNS 사용 컨테이너 네트워크를 생성합니다. `10.81.0.0/16` 서브넷 및 `10.81.0.1`을 게이트웨이로 사용합니다.

```
[podmgr@serverb ~]$ podman network create --gateway 10.81.0.1 \
--subnet 10.81.0.0/16 production
production
```

- 3.2. DNS 기능이 `production` 네트워크에서 활성화되어 있는지 확인합니다.

```
[podmgr@serverb ~]$ podman network inspect production
[
    {
        "name": "production",
        ...output omitted...
        "subnets": [
            {
                "subnet": "10.81.0.0/16",
                "gateway": "10.81.0.1"
            }
        ],
        ...output omitted...
        "dns_enabled": true,
        ...output omitted...
    }
]
```

4. `production` 네트워크에서 태그 번호가 가장 낮은 `registry.lab.example.com/rhel8/mariadb-103` 컨테이너 이미지를 기반으로 분리된 컨테이너 `db-app01`을 생성합니다. `/home/podmgr/storage/database` 디렉터리를 `db-app01` 컨테이너의 `/var/lib/mysql/data` 디렉터리를 위한 영구저장장치로 사용합니다. 로컬 시스템의 13306 포트를 컨테이너의 3306 포트에 맵핑합니다. 다음 표에 있는 값을 사용하여 컨테이너화된 데이터베이스를 생성하도록 환경 변수를 설정합니다.

변수	값
MYSQL_USER	developer
MYSQL_PASSWORD	redhat
MYSQL_DATABASE	inventory
MYSQL_ROOT_PASSWORD	redhat

- 4.1. `registry.lab.example.com/rhel8/mariadb` 컨테이너 이미지의 가장 오래된 버전 태그 번호를 검색합니다.

```
[podmgr@serverb ~]$ skopeo inspect \
docker://registry.lab.example.com/rhel8/mariadb-103
{
  "Name": "registry.lab.example.com/rhel8/mariadb-103",
  "Digest":
  "sha256:a95b678e52bb9f4305cb696e45c91a38c19a7c2c5c360ba6c681b10717394816",
  "RepoTags": [
    "1-86",
    "1-102",
    "latest"
  ...
  ...output omitted...
```

- 4.2. 이전 단계의 출력에서 가장 오래된 버전 태그 번호를 사용하여 `production` 네트워크에서 분리된 `db-app01` 컨테이너를 생성합니다. `/home/podmgr/storage/database` 디렉터리를 컨테이너의 영구저장장치로 사용합니다. 13306 포트를 3306 컨테이너 포트에 매핑합니다. 표에 있는 데이터를 사용하여 컨테이너의 환경 변수를 설정합니다.

```
[podmgr@serverb ~]$ podman run -d --name db-app01 \
-e MYSQL_USER=developer \
-e MYSQL_PASSWORD=redhat \
-e MYSQL_DATABASE=inventory \
-e MYSQL_ROOT_PASSWORD=redhat \
--network production -p 13306:3306 \
-v /home/podmgr/storage/database:/var/lib/mysql/data:Z \
registry.lab.example.com/rhel8/mariadb-103:1-86
...
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED
           STATUS PORTS NAMES
ba398d080e00 registry.lab.example.com/rhel8/mariadb-103:1-86 run-mysqld 20
           seconds ago Up 20 seconds ago 0.0.0.0:13306->3306/tcp db-app01
```

5. `db-app01` 컨테이너를 관리할 `systemd` 서비스 파일을 생성합니다. 서비스를 시작할 때 `systemd` 데몬이 원본 컨테이너를 유지하도록 `systemd` 서비스를 구성합니다. 컨테이너를 `systemd` 서비스로 시작하고 활성화합니다. 시스템 부팅 시 시작하도록 `db-app01` 컨테이너를 구성합니다.

- 5.1. 컨테이너 유닛 파일에 사용할 `~/.config/systemd/user/` 디렉터리를 생성합니다.

```
[podmgr@serverb ~]$ mkdir -p ~/.config/systemd/user/
```

- 5.2. db-app01 컨테이너에 대한 systemd 유닛 파일을 생성하고 유닛 파일을 ~/.config/systemd/user/ 디렉터리로 이동합니다.

```
[podmgr@serverb ~]$ podman generate systemd --name db-app01 --files
/home/podmgr/container-db-app01.service
[podmgr@serverb ~]$ mv container-db-app01.service ~/.config/systemd/user/
```

- 5.3. db-app01 컨테이너를 중지합니다.

```
[podmgr@serverb ~]$ podman stop db-app01
db-app01
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED
      STATUS PORTS NAMES
ba398d080e00 registry.lab.example.com/rhel8/mariadb-103:1-86 run-mysqld About
an hour ago Exited (0) 3 seconds ago 0.0.0.0:13306->3306/tcp db-app01
```

- 5.4. 사용자 systemd 서비스를 다시 로드하여 새 서비스 유닛을 사용합니다.

```
[podmgr@serverb ~]$ systemctl --user daemon-reload
```

- 5.5. db-app01 컨테이너의 systemd 유닛을 시작하고 활성화합니다.

```
[podmgr@serverb ~]$ systemctl --user enable --now container-db-app01
Created symlink /home/podmgr/.config/systemd/user/default.target.wants/container-
db-app01.service → /home/podmgr/.config/systemd/user/container-db-app01.service.
[podmgr@serverb ~]$ systemctl --user status container-db-app01
● container-db-app01.service - Podman container-db-app01.service
   Loaded: loaded (/home/podmgr/.config/systemd/user/container-db-app01.service;
   disabled; vendor preset: disabled)
     Active: active (running) since Tue 2022-05-10 22:16:23 EDT; 7s ago
...output omitted...
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID IMAGE COMMAND CREATED
      STATUS PORTS NAMES
ba398d080e00 registry.lab.example.com/rhel8/mariadb-103:1-86 run-mysqld 59
seconds ago Up About a minute ago 0.0.0.0:13306->3306/tcp db-app01
```

- 5.6. loginctl 명령을 사용하여 시스템 부팅 시 시작하도록 db-app01 컨테이너를 구성합니다.

```
[podmgr@serverb ~]$ loginctl enable-linger
```

6. /home/podmgr/db-dev/inventory.sql 스크립트를 db-app01 컨테이너의 /tmp 디렉터리에 복사하고 스크립트를 컨테이너 내부에서 실행합니다. 스크립트를 로컬로 실행한 경우 mysql -u root inventory < /tmp/inventory.sql 명령을 사용합니다.

- 6.1. /home/podmgr/db-dev/inventory.sql 스크립트를 db-app01 컨테이너의 /tmp 디렉터리에 복사합니다.

```
[podmgr@serverb ~]$ podman cp /home/podmgr/db-dev/inventory.sql \
db-app01:/tmp/inventory.sql
```

- 6.2. db-app01 컨테이너에서 inventory.sql 스크립트를 실행합니다.

```
[podmgr@serverb ~]$ podman exec -it db-app01 sh -c 'mysql -u root inventory
< /tmp/inventory.sql'
```

7. /home/podmgr/http-dev 디렉터리의 컨테이너 파일을 사용하여 production 네트워크에서 분리된 컨테이너 http-app01을 생성합니다. 컨테이너 이미지 이름은 9.0 태그가 있는 http-client여야 합니다. 로컬 시스템의 8080 포트를 컨테이너의 8080 포트에 매핑합니다.

- 7.1. /home/podmgr/http-dev 디렉터리에 컨테이너 파일이 포함된 http-client:9.0 이미지를 생성합니다.

```
[podmgr@serverb ~]$ podman build -t http-client:9.0 http-dev/
STEP 1/7: FROM registry.lab.example.com/rhel8/php-74:1-63
...output omitted...
```

- 7.2. production 네트워크에 분리된 컨테이너 http-app01을 생성합니다. 로컬 시스템에서 8080 포트를 컨테이너의 8080 포트에 매핑합니다.

```
[podmgr@serverb ~]$ podman run -d --name http-app01 \
--network production -p 8080:8080 localhost/http-client:9.0
[podmgr@serverb ~]$ podman ps -a
CONTAINER ID  IMAGE                                COMMAND      CREATED
              STATUS     PORTS          NAMES
ba398d080e00  registry.lab.example.com/rhel8/mariadb-103:1-86  run-mysqld  20
              minutes ago Up 20 seconds ago  0.0.0.0:13306->3306/tcp  db-app01
ee424df19621  localhost/http-client:9.0                  /bin/sh -c    4
              seconds ago Up 4 seconds ago   0.0.0.0:8080->8080/tcp  http-app01
```

8. http-app01 컨테이너의 컨텐츠를 쿼리합니다. 클라이언트의 컨테이너 이름이 표시되고 데이터베이스 상태가 'up'인지 확인합니다.

- 8.1. http-app01 컨테이너가 http 요청에 응답하는지 확인합니다.

```
[podmgr@serverb ~]$ curl 127.0.0.1:8080
This is the server http-app01 and the database is up
```

9. workstation 시스템에 student 사용자로 돌아갑니다.

```
[podmgr@serverb ~]$ exit
logout
Connection to serverb closed.
[student@workstation ~]$
```

평가

workstation 시스템에서 student 사용자로 lab 명령을 사용하여 작업을 평가합니다. 보고된 오류를 모두 수정하고 성공할 때까지 명령을 다시 실행합니다.

```
[student@workstation ~]$ lab grade rhcsa-compreview4
```

완료

workstation 시스템에서 student 사용자 홈 디렉터리로 변경하고 lab 명령을 사용하여 이 연습을 완료합니다. 이전 연습 문제의 리소스가 다음 연습 문제에 영향을 미치지 않도록 하려면 이 단계가 중요합니다.

```
[student@workstation ~]$ lab finish rhcsa-compreview4
```

이것으로 섹션을 완료합니다.