



Red Hat Enterprise Linux 9

RHEL에서 시스템 역할을 사용하여 관리 및 구성 작업

Red Hat Ansible Automation Platform 플레이북을 사용하여 RHEL 시스템 역할 적용

Red Hat Enterprise Linux 9 RHEL에서 시스템 역할을 사용하여 관리 및 구성 작업

Red Hat Ansible Automation Platform 플레이북을 사용하여 RHEL 시스템 역할 적용

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

법적 공지

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Administration_and_configuration_tasks_using_System_Roles_in_RHEL.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution-Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered 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 and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. 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 an official 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 OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

초록

이 문서에서는 Red Hat Enterprise Linux 9에서 Ansible을 사용하여 시스템 역할 구성에 대해 설명합니다. RHEL 시스템 역할은 Red Hat Enterprise Linux를 관리하고 구성하기 위한 안정적이고 일관된 구성 인터페이스를 제공하는 Ansible 역할, 모듈 및 플레이북의 컬렉션입니다. RHEL 9의 여러 주요 릴리스 버전과 호환되도록 설계되었습니다.

차례

보다 포괄적 수용을 위한 오픈 소스 용어 교체	6
RED HAT 문서에 관한 피드백 제공	7
1장. RHEL 시스템 역할 시작하기	8
1.1. RHEL 시스템 역할 소개	8
1.2. RHEL 시스템 역할 용어	8
1.3. 역할 적용	9
1.4. 추가 리소스	11
2장. RHEL 시스템 역할 설치	13
2.1. 시스템에서 RHEL 시스템 역할 설치	13
3장. RHEL 시스템 역할에 대한 자동화를 활성화하도록 패키지 업데이트	14
3.1. ANSIBLE ENGINE와 ANSIBLE CORE 간의 차이점	14
3.2. ANSIBLE ENGINE에서 ANSIBLE CORE로 마이그레이션	15
4장. 컬렉션 설치 및 사용	17
4.1. ANSIBLE 컬렉션 소개	17
4.2. 컬렉션 구조	17
4.3. CLI를 사용하여 컬렉션 설치	18
4.4. AUTOMATION HUB에서 컬렉션 설치	20
4.5. 컬렉션을 사용하여 RHEL 시스템 역할을 기록하는 터미널 세션 배포	21
5장. RHEL의 ANSIBLE IPMI 모듈	24
5.1. RHEL_BACKEND 컬렉션	24
5.2. CLI를 사용하여 RHEL RWX 컬렉션 설치	25
5.3. IPMI_BOOT 모듈 사용 예	26
5.4. IPMI_POWER 모듈 사용 예	27
6장. ANSIBLE 역할을 사용하여 커널 매개 변수를 영구적으로 구성	29
6.1. 커널 설정 역할 소개	29
6.2. 커널 설정 역할을 사용하여 선택한 커널 매개변수 적용	30
7장. RHEL 시스템 역할을 사용하여 네트워크 연결 구성	35
7.1. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성	35
7.2. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성	37
7.3. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성	40
7.4. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성	41
7.5. RHEL 시스템 역할을 사용하여 VLAN 태그 지정 구성	43
7.6. RHEL 시스템 역할을 사용하여 네트워크 브리지 구성	46
7.7. RHEL 시스템 역할을 사용하여 네트워크 본딩 구성	48
7.8. RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성	50
7.9. SYSTEM ROLES를 사용하여 기존 연결에 기본 게이트웨이 설정	53
7.10. RHEL 시스템 역할을 사용하여 정적 경로 구성	56
7.11. RHEL 시스템 역할을 사용하여 ETHTOOL 기능 설정	59
7.12. RHEL 시스템 역할을 사용하여 ETHTOOL COALESCE 설정 설정	62
8장. 시스템 역할의 POSTFIX 역할 변수	66
8.1. 추가 리소스	67
9장. 시스템 역할을 사용하여 SELINUX 구성	68
9.1. SELINUX 시스템 역할 소개	68
9.2. SELINUX 시스템 역할을 사용하여 여러 시스템에 SELINUX 설정 적용	69

10장. 로깅 시스템 역할 사용	72
10.1. 로깅 시스템 역할	72
10.2. 로깅 시스템 역할 매개변수	73
10.3. 로컬 로깅 시스템 역할 적용	75
10.4. 로컬 로깅 시스템 역할에서 로그 필터링	77
10.5. 로깅 시스템 역할을 사용하여 원격 로깅 솔루션 적용	80
10.6. TLS를 사용하여 로깅 시스템 역할 사용	84
10.6.1. TLS로 클라이언트 로깅 구성	84
10.6.2. TLS를 사용하여 서버 로깅 구성	86
10.7. RELP에서 로깅 시스템 역할 사용	89
10.7.1. RELP로 클라이언트 로깅 구성	89
10.7.2. RELP로 서버 로깅 구성	92
10.8. 추가 리소스	94
11장. SSH 시스템 역할을 사용하여 보안 통신 구성	96
11.1. SSH SERVER 시스템 역할 변수	96
11.2. SSH 서버 시스템 역할을 사용하여 OPENSSH 서버 구성	99
11.3. SSH 클라이언트 시스템 역할 변수	103
11.4. SSH 클라이언트 시스템 역할을 사용하여 OPENSSH 클라이언트 구성	105
11.5. 비독점 구성에 대해 SSH 서버 시스템 역할 사용	108
12장. VPN RHEL 시스템 역할을 사용하여 IPSEC을 사용하여 VPN 연결 구성	111
12.1. VPN 시스템 역할을 사용하여 IPSEC으로 호스트 간 VPN 생성	111
12.2. VPN 시스템 역할을 사용하여 IPSEC을 통한 OPPORTUNISTIC 메시 VPN 연결 생성	114
12.3. 추가 리소스	117
13장. 시스템 전체에서 사용자 지정 암호화 정책 설정	118
13.1. 암호화 정책 시스템 역할 변수 및 사실	118
13.2. 암호화 정책 시스템 역할을 사용하여 사용자 정의 암호화 정책 설정	119
13.3. 추가 리소스	121
14장. CLEVIS 및 TANG 시스템 역할 사용	122
14.1. CLEVIS 및 TANG 시스템 역할 소개	122
14.2. 여러 TANG 서버를 설정하는 데 CLEVIS 서버 시스템 역할 사용	123
14.3. 여러 CLEVIS 클라이언트를 설정하는 데 CLEVIS 클라이언트 시스템 역할 사용	125
15장. RHEL 시스템 역할을 사용하여 인증서 요청	128
15.1. 인증서 문제 및 업데이트 시스템 역할	128
15.2. 인증서 문제 및 갱신 시스템 역할을 사용하여 새 자체 서명 인증서 요청	129
15.3. 인증서 문제 및 갱신 시스템 역할을 사용하여 IDM CA에서 새 인증서 요청	131
15.4. 인증서 문제 및 갱신 시스템 역할을 사용하여 인증서 발급 후 실행할 명령 지정	133
16장. RHEL 시스템 역할을 사용하여 KDUMP 구성	136
16.1. 커널 덤프 RHEL 시스템 역할	136
16.2. 커널 덤프 역할 매개변수	136
16.3. RHEL 시스템 역할을 사용하여 KDUMP 구성	136
17장. RHEL 시스템 역할을 사용하여 로컬 스토리지 관리	139
17.1. 스토리지 역할 소개	139
17.2. 스토리지 시스템 역할에서 스토리지 장치를 식별하는 매개변수	140
17.3. 블록 장치에 XFS 파일 시스템을 생성하는 ANSIBLE 플레이북의 예	140
17.4. ANSIBLE 플레이북의 예제로 파일 시스템을 지속적으로 마운트합니다.	142
17.5. 논리 볼륨을 관리하는 ANSIBLE 플레이북 예	142
17.6. 온라인 블록 삭제 활성화를 위한 예제 ANSIBLE 플레이북	143
17.7. EXT4 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예	144

17.8. EXT3 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예	145
17.9. 스토리지 RHEL 시스템 역할을 사용하여 기존 EXT4 또는 EXT3 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북의 예	146
17.10. 스토리지 RHEL 시스템 역할을 사용하여 LVM의 기존 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북 예	147
17.11. 스토리지 RHEL 시스템 역할을 사용하여 스왑 파티션을 생성하는 ANSIBLE 플레이북 예	148
17.12. 스토리지 시스템 역할을 사용하여 RAID 볼륨 구성	149
17.13. 스토리지 시스템 역할을 사용하여 RAID로 LVM 풀 구성	150
17.14. 스토리지 RHEL 시스템 역할을 사용하여 LVM에서 VDO 볼륨을 압축하고 중복하기 위한 ANSIBLE 플레이북 예	151
17.15. 스토리지 시스템 역할을 사용하여 LUKS 암호화된 볼륨 생성	152
17.16. 스토리지 RHEL 시스템 역할을 사용하여 풀 볼륨 크기를 백분율로 표시하는 ANSIBLE 플레이북의 예	154
17.17. 추가 리소스	155
18장. RHEL 시스템 역할을 사용하여 시간 동기화 구성	156
18.1. 시간 동기화 시스템 역할	156
18.2. 단일 서버 풀에 대한 시간 동기화 시스템 역할 적용	156
18.3. 클라이언트 서버에 시간 동기화 시스템 역할 적용	157
18.4. 시간 동기화 시스템 역할 변수	159
19장. RHEL 시스템 역할을 사용하여 성능 모니터링	161
19.1. 지표 시스템 역할 소개	161
19.2. 시각화로 로컬 시스템을 모니터링하기 위해 메트릭 시스템 역할을 사용	162
19.3. 지표 시스템 역할을 사용하여 자신을 모니터링하기 위해 개별 시스템 세트를 설정	163
19.4. 메트릭 시스템 역할을 사용하여 로컬 시스템을 통해 중앙 집중식으로 시스템 그룹을 모니터링할 수 있습니다.	164
19.5. METRICS 시스템 역할을 사용하여 시스템을 모니터링하는 동안 인증 설정	165
19.6. METRICS 시스템 역할을 사용하여 SQL SERVER에 대한 메트릭 컬렉션을 구성하고 활성화합니다.	166
20장. MICROSOFT SQL SERVER ANSIBLE 역할을 사용하여 MICROSOFT SQL SERVER 구성	168
20.1. 사전 요구 사항	168
20.2. MICROSOFT SQL SERVER ANSIBLE 역할 설치	168
20.3. MICROSOFT SQL SERVER ANSIBLE 역할을 사용하여 SQL SERVER 설치 및 구성	169
20.4. TLS 변수	170
20.5. MLSERVICES에 대한 EULA 수락	171
20.6. MICROSOFT ODBC 17에 대한 EULA 수락	171
21장. 터미널 세션 기록 RHEL 시스템 역할을 사용하여 세션 기록 시스템 구성	173
21.1. 터미널 세션 시스템 역할 기록	173
21.2. 터미널 세션 기록 시스템 역할의 구성 요소 및 매개변수	173
21.3. RHEL 시스템 역할 기록 터미널 세션 배포	174
21.4. 그룹 또는 사용자 목록을 제외하고 RHEL 시스템 역할을 기록하는 터미널 세션 기록	176
21.5. CLI에서 배포된 터미널 세션 기록 시스템 역할을 사용하여 세션 기록	178
21.6. CLI를 사용하여 기록된 세션 모니터링	180
22장. 시스템 역할을 사용하여 고가용성 클러스터 구성	182
22.1. HA 클러스터 시스템 역할 변수	182
22.2. HA 클러스터 시스템 역할에 대한 인벤토리 지정	199
22.3. 리소스 없이 실행되는 고가용성 클러스터 구성	200
22.4. 펜싱 및 리소스를 사용하여 고가용성 클러스터 구성	202
22.5. 리소스 제약 조건을 사용하여 고가용성 클러스터 구성	205
22.6. HA 클러스터 시스템 역할을 사용하여 고가용성 클러스터에서 APACHE HTTP 서버 구성	208
22.7. 추가 리소스	213
23장. COCKPIT RHEL 시스템 역할을 사용하여 웹 콘솔 설치 및 구성	214

23.1. COCKPIT 시스템 역할	214
23.2. COCKPIT RHEL 시스템 역할에 대한 변수	214
23.3. COCKPIT RHEL 시스템 역할을 사용하여 웹 콘솔 설치	215
23.4. 인증서 RHEL 시스템 역할을 사용하여 새 인증서 설정	216

보다 포괄적 수용을 위한 오픈 소스 용어 교체

Red Hat은 코드, 문서, 웹 속성에서 문제가 있는 용어를 교체하기 위해 최선을 다하고 있습니다. 먼저 마스터(master), 슬레이브(slave), 블랙리스트(blacklist), 화이트리스트(whitelist) 등 네 가지 용어를 교체하고 있습니다. 이러한 변경 작업은 작업 범위가 크므로 향후 여러 릴리스에 걸쳐 점차 구현할 예정입니다. 자세한 내용은 [CTO Chris Wright의 메시지](#)를 참조하십시오.

RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견을 보내 주십시오. 문서를 개선할 수 있는 방법에 대해 알려주십시오.

- 특정 문구에 대한 간단한 의견 작성 방법은 다음과 같습니다.
 1. 문서가 *Multi-page HTML* 형식으로 표시되는지 확인합니다. 또한 문서 오른쪽 상단에 **피드백** 버튼이 있는지 확인합니다.
 2. 마우스 커서를 사용하여 주석 처리하려는 텍스트 부분을 강조 표시합니다.
 3. 강조 표시된 텍스트 아래에 표시되는 **피드백 추가** 팝업을 클릭합니다.
 4. 표시된 지침을 따릅니다.
- Bugzilla를 통해 피드백을 제출하려면 새 티켓을 생성하십시오.
 1. [Bugzilla](#) 웹 사이트로 이동하십시오.
 2. 구성 요소로 **문서**를 사용합니다.
 3. **설명** 필드에 문서 개선을 위한 제안 사항을 기입하십시오. 관련된 문서의 해당 부분 링크를 알려주십시오.
 4. **버그 제출**을 클릭합니다.

1장. RHEL 시스템 역할 시작하기

이 섹션에서는 RHEL 시스템 역할에 대해 설명합니다. 또한 Ansible 플레이북을 통해 특정 역할을 적용하여 다양한 시스템 관리 작업을 수행하는 방법을 설명합니다.

1.1. RHEL 시스템 역할 소개

RHEL 시스템 역할은 Ansible 역할 및 모듈의 컬렉션입니다. RHEL 시스템 역할은 여러 RHEL 시스템을 원격으로 관리하는 구성 인터페이스를 제공합니다. 이 인터페이스를 사용하면 여러 버전의 RHEL에서 시스템 구성을 관리하고 새로운 주요 릴리스를 채택할 수 있습니다.

Red Hat Enterprise Linux 9에서 인터페이스는 현재 다음 역할로 구성됩니다.

- 인증서 문제 및 업데이트
- 커널 설정
- 지표
- 네트워크 Bound 디스크 암호화 클라이언트 및 네트워크 Bound Disk Encryption 서버
- 네트워킹
- nsb
- SSH 클라이언트
- SSH 서버
- 시스템 전체 암호화 정책
- 터미널 세션 기록

이러한 모든 역할은 **AppStream** 리포지토리에서 사용할 수 있는 **rhel-system-roles** 패키지에서 제공됩니다.

추가 리소스

- [Red Hat Enterprise Linux \(RHEL\) 시스템 역할](#)
- `/usr/share/doc/rhel-system-roles/` 디렉터리에 있는 문서 ^[1]

1.2. RHEL 시스템 역할 용어

이 문서에서 다음 용어를 찾을 수 있습니다.

Ansible 플레이북

플레이북은 Ansible의 구성, 배포 및 오케스트레이션 언어입니다. 원격 시스템을 적용하려는 정책이나 일반 IT 프로세스에서 일련의 단계를 설명할 수 있습니다.

제어 노드

Ansible이 설치된 모든 시스템. 모든 제어 노드에서 `/usr/bin/ansible` 또는 `/usr/bin/ansible-playbook`을 호출하는 명령과 플레이북을 실행할 수 있습니다. Python이 설치된 모든 컴퓨터를 제어 노드로 사용할 수 있습니다. 즉 랩탑, 공유 데스크탑 및 서버는 모두 Ansible을 실행할 수 있습니다. 그러나 Windows 머신을 제어 노드로 사용할 수 없습니다. 여러 컨트롤 노드를 사용할 수 있습니다.

인벤토리

관리형 노드 목록입니다. 인벤토리 파일을 "hostfile"이라고도 합니다. 인벤토리는 각 관리 노드에 대해 IP 주소와 같은 정보를 지정할 수 있습니다. 인벤토리는 보다 쉽게 스케일링할 수 있도록 관리형 노드를 구성하고, 그룹을 생성 및 중첩할 수 있습니다. 인벤토리에 대한 자세한 내용은 Working with Inventory 섹션을 참조하십시오.

관리형 노드

Ansible로 관리하는 네트워크 장치, 서버 또는 둘 다입니다. 관리형 노드를 "hosts"라고도 합니다. Ansible은 관리형 노드에 설치되지 않습니다.

1.3. 역할 적용

다음 절차에서는 특정 역할을 적용하는 방법을 설명합니다.

사전 요구 사항

- **rhel-system-roles** 패키지가 제어 노드로 사용하려는 시스템에 설치되어 있는지 확인합니다.

```
# dnf install rhel-system-roles
```

1. Ansible Core 패키지를 설치합니다.

```
# dnf install ansible-core
```

Ansible Core 패키지는 **ansible-playbook** CLI, Ansible Vault 기능 및 RHEL Ansible 콘텐츠에 필요한 기본 모듈 및 필터를 제공합니다.

- Ansible 인벤토리를 생성할 수 있는지 확인합니다.
인벤토리는 Ansible 플레이북에서 사용하는 호스트, 호스트 그룹 및 일부 구성 매개변수를 나타냅니다.

플레이북은 일반적으로 사람이 읽을 수 있으며 **ini**, **yaml**, **json** 및 기타 파일 형식으로 정의됩니다.

- Ansible 플레이북을 생성할 수 있는지 확인합니다.
플레이북은 Ansible의 구성, 배포 및 오케스트레이션 언어를 나타냅니다. 플레이북을 사용하면 원격 시스템의 구성을 선언 및 관리하고 여러 원격 시스템을 배포하거나 수동 순서가 지정된 프로세스의 단계를 오케스트레이션할 수 있습니다.

플레이북은 하나 이상의 **플레이** 목록입니다. 모든 **플레이**에는 Ansible 변수, 작업 또는 역할이 포함될 수 있습니다.

플레이북은 사람이 읽을 수 있으며 **yaml** 형식으로 정의됩니다.

절차

1. 관리하려는 호스트 및 그룹을 포함하는 필수 Ansible 인벤토리를 생성합니다. 다음과 같습니다.
inventory.ini라는 호스트 그룹의 **inventory.ini** 파일을 사용하는 예입니다.

```
[webserver]
host1
host2
host3
```

2.

필요한 역할을 포함하여 **Ansible** 플레이북을 생성합니다. 다음 예는 플레이북에 **roles:** 옵션을 통해 역할을 사용하는 방법을 보여줍니다.

다음 예제에서는 지정된 플레이에 대해 **roles:** 옵션을 통해 역할을 사용하는 방법을 보여줍니다.

```
---
- hosts: webserver
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

참고

모든 역할에는 **README** 파일이 포함되어 있으며, 이 파일은 역할 및 지원되는 매개 변수 값을 사용하는 방법을 설명합니다. 역할의 문서 디렉터리에서 특정 역할에 대한 예제 플레이북을 찾을 수도 있습니다. 이러한 문서 디렉터리는 기본적으로 **rhel-system-roles** 패키지와 함께 제공되며 다음 위치에서 확인할 수 있습니다.

```
/usr/share/doc/rhel-system-roles/SUBSYSTEM/
```

SUBSYSTEM을 **postfix,metrics,network,tlog** 또는 **ssh** 와 같은 필수 역할의 이름으로 바꿉니다.

3.

특정 호스트에서 플레이북을 실행하려면 다음 중 하나를 수행해야 합니다.

- **hosts: host1[,host2,...]**을 사용하도록 플레이북을 편집합니다.], 또는 호스트: **all**, and **execute the command:**

```
# ansible-playbook name.of.the.playbook
```

- 인벤토리를 편집하여 사용하려는 호스트가 그룹에 정의되어 있는지 확인하고 명령을 실행합니다.

```
# ansible-playbook -i name.of.the.inventory name.of.the.playbook
```

- **ansible-playbook** 명령을 실행할 때 모든 호스트를 지정합니다.

```
# ansible-playbook -i host1,host2,... name.of.the.playbook
```

중요

-i 플래그는 사용 가능한 모든 호스트의 인벤토리를 지정합니다. 대상 호스트가 여러 개 있지만 플레이북을 실행할 호스트를 선택하려면 플레이북에서 변수를 추가하여 호스트를 선택할 수 있습니다. 예를 들면 다음과 같습니다.

Ansible Playbook | example-playbook.yml:

```
- hosts: "{{ target_host }}"
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.postfix
```

Playbook 실행 명령:

```
# ansible-playbook -i host1,..hostn -e target_host=host5 example-playbook.yml
```

추가 리소스

- [Ansible Playbook](#)

- [Ansible 플레이북에서 역할 사용](#)

- [Ansible 플레이북의 예](#)

- [인벤토리를 만들고 작업하는 방법은 무엇입니까?](#)

- [ansible-playbook 툴](#)

1.4. 추가 리소스

- **Red Hat Enterprise Linux (RHEL) 시스템 역할**
- **RHEL 시스템 역할을 사용하여 로컬 스토리지 관리**
- **RHEL 시스템 역할을 사용하여 여러 시스템에 동일한 SELinux 구성 배포**

[1]

이 문서는 **rhel-system-roles** 패키지를 사용하여 자동으로 설치됩니다.

2장. RHEL 시스템 역할 설치

시스템 역할을 사용하기 전에 시스템에 설치해야 합니다.

2.1. 시스템에서 RHEL 시스템 역할 설치

RHEL 시스템 역할을 사용하려면 시스템에 필수 패키지를 설치합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 제어 노드로 사용하려는 시스템에 **Ansible** 패키지가 설치되어 있어야 합니다.

절차

1. 제어 노드로 사용하려는 시스템에 **rhel-system-roles** 패키지를 설치합니다.

```
# dnf install rhel-system-roles
```

2. **Ansible Core** 패키지를 설치합니다.

```
# dnf install ansible-core
```

Ansible Core 패키지는 **ansible-playbook CLI**, **Ansible Vault** 기능 및 **RHEL Ansible** 콘텐츠에 필요한 기본 모듈 및 필터를 제공합니다.

결과적으로 **Ansible** 플레이북을 생성할 수 있습니다.

추가 리소스

- [RHEL\(Red Hat Enterprise Linux\) 시스템 역할](#)
- [ansible-playbook](#) 도움말 페이지.

3장. RHEL 시스템 역할에 대한 자동화를 활성화하도록 패키지 업데이트

RHEL 9.0 GA 릴리스에서 **Ansible Engine**은 더 이상 지원되지 않습니다. 대신 이 및 향후 RHEL 릴리스에는 **Ansible Core**가 포함되어 있습니다.

RHEL 9.0 GA의 **Ansible Core**를 사용하여 Red Hat 제품에서 작성 또는 생성한 **Ansible** 자동화 콘텐츠를 활성화할 수 있습니다.

Ansible Core에는 **ansible-playbook** 및 **ansible** 명령과 같은 **Ansible** 명령줄 툴과 작은 **기본 제공 Ansible 플러그인** 세트가 포함되어 있습니다.

3.1. ANSIBLE ENGINE과 ANSIBLE CORE 간의 차이점

RHEL 8.5 및 이전 버전에서 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 액세스하여 **Ansible** 기반 자동화를 Red Hat 시스템에 사용할 수 있었습니다.

Ansible 서브스크립션 없이 **Ansible Engine**을 사용하는 경우 지원 범위는 RHEL 시스템 역할, **Insights** 수정 플레이북, **OpenSCAP Ansible** 수정 플레이북과 같은 Red Hat 제품에서 생성 또는 생성하는 **Ansible** 플레이북으로 제한됩니다.

RHEL 8.6 이상 버전에서 **Ansible Core**는 **Ansible Engine**을 대체합니다. **ansible-core** 패키지는 Red Hat에서 제공하는 자동화 콘텐츠를 활성화하기 위해 RHEL 9 AppStream 리포지토리에 포함되어 있습니다. RHEL에서 **Ansible Core**에 대한 지원 범위는 이전 RHEL 버전과 동일하게 유지됩니다.

- 지원은 RHEL 시스템 역할과 같이 Red Hat 제품에 포함되거나 Insights에서 생성한 플레이북을 수정하는데 포함된 모든 **Ansible** 플레이북, 역할, 모듈로 제한됩니다.
- **Ansible Core**를 사용하면 RHEL 시스템 역할 및 Insights 해결 플레이북과 같이 지원되는 RHEL **Ansible** 콘텐츠의 모든 기능을 사용할 수 있습니다.

Ansible Engine 리포지토리는 RHEL 8.6;에서 계속 사용할 수 있지만 보안 또는 버그 수정 업데이트가 제공되지 않으며 RHEL 8.6 이상에 포함된 **Ansible** 자동화 콘텐츠와 호환되지 않을 수 있습니다.

기본 플랫폼 및 코어 유지 관리 모듈에 대한 추가 지원을 받으려면 **Ansible Automation Platform** 서브스크립션이 필요합니다.

추가 리소스

- [RHEL에서 Ansible Core에 대한 지원 범위](#)

3.2. ANSIBLE ENGINE에서 ANSIBLE CORE로 마이그레이션

사전 요구 사항

- RHEL 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 *관리형 노드*에 대한 액세스 및 권한.
 - 관리 노드를 나열하는 인벤토리 파일.

절차

- Ansible Engine 설치 제거:**

```
# dnf remove ansible
```
- ansible-2-for-rhel-8-x86_64-rpms 리포지토리를 비활성화합니다.**

```
# subscription-manager repos --disable ansible-2-for-rhel-8-x86_64-rpms
```
- RHEL 8 AppStream 리포지토리에서 사용할 수 있는 Ansible Core를 설치합니다.**

```
# dnf install ansible-core
```

검증

- 시스템에 **ansible-core** 패키지가 있는지 확인합니다.


```
# dnf info ansible-core
```

ansible-core 패키지가 실제로 시스템에 있는 경우 명령 출력은 패키지 이름, 버전, 릴리스, 크기 등에 대한 정보를 표시합니다.

```
Available Packages
```

Name : ansible-core
Version : 2.12.2
Release : 1.fc34
Architecture : noarch
Size : 2.4 M
Source : ansible-core-2.12.2-1.fc34.src.rpm
Repository : updates
Summary : A radically simple IT automation system
URL : http://ansible.com

추가 리소스

- [RHEL 9에서 Ansible 사용.](#)

4장. 컬렉션 설치 및 사용

4.1. ANSIBLE 컬렉션 소개

Ansible Collections는 자동화를 배포, 유지 관리 및 사용하는 새로운 방법입니다. 플레이북, 역할, 모듈 및 플러그인과 같은 여러 유형의 **Ansible** 콘텐츠를 결합하면 유연성 및 확장성 개선의 이점을 누릴 수 있습니다.

Ansible 컬렉션은 기존 **RHEL** 시스템 역할 형식의 옵션입니다. **Ansible** 컬렉션 형식에서 **RHEL** 시스템 역할을 사용하는 것은 기존 **RHEL** 시스템 역할 형식에서 사용하는 것과 거의 동일합니다. 차이점은 **Ansible Collections**가 네임스페이스 및 컬렉션 이름으로 구성된 정규화된 컬렉션 이름 (FQCN) 개념을 사용한다는 것입니다. 사용하는 네임스페이스는 **redhat** 이고 컬렉션 이름은 **rhel_system_roles** 입니다. 따라서 커널 설정 역할에 대한 기존 **RHEL** 시스템 역할 형식이 **rhel-system-roles.kernel_settings** 로 표시되지만, 커널 설정 역할에 대해 정규화된 컬렉션 이름을 사용하여 **redhat.rhel_system_roles.kernel_settings**.

네임스페이스와 컬렉션 이름을 결합하면 해당 오브젝트가 고유합니다. 또한 충돌 없이 **Ansible** 컬렉션 및 네임스페이스에서 오브젝트를 공유할 수 있습니다.

추가 리소스

- [Automation Hub](#) 에 액세스하여 **Red Hat Certified Collections**를 사용하려면 **Ansible Automation Platform(AAP** 서브스크립션)이 있어야 합니다.

4.2. 컬렉션 구조

컬렉션은 **Ansible** 콘텐츠의 패키지 형식입니다. 데이터 구조는 다음과 같습니다.

- **docs/**: 역할에서 문서를 제공하는 경우 예제와 함께 컬렉션에 대한 로컬 문서
- **Galaxy.yml**: **Ansible Collection** 패키지에 포함될 **MANIFEST.json**의 소스 데이터
- **playbooks/**: **Playbook**은 여기에서 사용할 수 있습니다.
 - **tasks/**: **include_tasks/import_tasks** 사용을 위한 'task list files'가 있습니다.

- **plugins/:** 모든 **Ansible** 플러그인 및 모듈은 여기에서 사용할 수 있습니다. 각각의 하위 디렉토리
 - **modules/:** **Ansible** 모듈
 - **modules_utils/:** 모듈 개발을 위한 공통 코드
 - **lookup/:** 플러그인 검색
 - **filter/:** **Jinja2** 필터 플러그인
 - **connection/:** 기본값을 사용하지 않는 경우 연결 플러그인이 필요합니다.
- **roles/:** **Ansible** 역할용 디렉토리
- **tests/:** 컬렉션의 콘텐츠에 대한 테스트

4.3. CLI를 사용하여 컬렉션 설치

컬렉션은 플레이북, 역할, 모듈 및 플러그인을 포함할 수 있는 **Ansible** 콘텐츠의 배포 형식입니다.

브라우저를 통해 또는 명령줄을 사용하여 **Ansible Galaxy**를 통해 **Collections**를 설치할 수 있습니다.

사전 요구 사항

- 하나 이상의 관리형 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.

절차

- **RPM** 패키지를 통해 컬렉션을 설치합니다.

```
# dnf install rhel-system-roles
```

설치가 완료되면 **redhat.rhel_system_roles.<role_name>** 으로 역할을 사용할 수 있습니다. 또한 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** 에서 각 역할에 대한 문서를 찾을 수 있습니다.

검증 단계

Collections가 성공적으로 설치되었는지 확인하려면 **localhost**에 **kernel_settings**를 적용할 수 있습니다.

1. **tests_default.yml** 중 하나를 작업 디렉터리에 복사합니다.

```
$ cp
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
tests_default.yml .
```

2. **"hosts: all"**을 **"hosts: localhost"**로 교체하여 로컬 시스템에서만 플레이북을 실행하도록 파일을 편집합니다.
3. 검사 모드에서 **ansible-playbook**을 실행합니다. 이 설정은 시스템의 설정을 변경하지 않습니다.

```
$ ansible-playbook --check tests_default.yml
```

이 명령은 **failed=0** 값을 반환합니다.

추가 리소스

- **ansible-playbook** 도움말 페이지.

4.4. AUTOMATION HUB에서 컬렉션 설치

Automation Hub를 사용하는 경우 **Automation Hub**에 호스팅된 **RHEL** 시스템 역할 컬렉션을 설치할 수 있습니다.

사전 요구 사항

- 하나 이상의 **관리형** 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 **제어** 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.

절차

1. **ansible.cfg** 구성 파일의 콘텐츠의 기본 소스로 **Red Hat Automation Hub**를 정의합니다. [컨텐츠의 기본 소스로 Red Hat Automation Hub 구성](#)을 참조하십시오.

2. **Automation Hub**에서 **redhat.rhel_system_roles** 컬렉션을 설치합니다.

```
# ansible-galaxy collection install redhat.rhel_system_roles
```

설치가 완료되면 **redhat.rhel_system_roles.<role_name>** 으로 역할을 사용할 수 있습니다. 또한 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** 에서 각 역할에 대한 문서를 찾을 수 있습니다.

검증 단계

컬렉션이 성공적으로 설치되었는지 확인하려면 **localhost**에 **kernel_settings**를 적용할 수 있습니다.

1. **tests_default.yml** 중 하나를 작업 디렉터리에 복사합니다.

```
$ cp
/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/tests/kernel_settings
ests_default.yml .
```

2. **"hosts: all"**을 **"hosts: localhost"**로 교체하여 로컬 시스템에서만 플레이북을 실행하도록 파일을 편집합니다.

3. 검사 모드에서 **ansible-playbook**을 실행합니다. 이 설정은 시스템의 설정을 변경하지 않습니다.

```
$ ansible-playbook --check tests_default.yml
```

이 명령은 **failed=0** 값을 사용하여 반환하는 것을 확인할 수 있습니다.

추가 리소스

- **ansible-playbook** 도움말 페이지.

4.5. 컬렉션을 사용하여 RHEL 시스템 역할을 기록하는 터미널 세션 배포

다음은 **Collections**를 사용하여 플레이북을 준비하고 적용하여 별도의 시스템 집합에 로깅 솔루션을 배포하는 예입니다.

사전 요구 사항

- **Galaxy** 컬렉션이 설치되어 있습니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
```

```
- name: Deploy session recording
  hosts: all
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recordeduser

  roles:
    - redhat.rhel-system-roles.tlog
```

여기서,

- **tlog_scope_sssd:**
 - 일부는 일부 또는 **none** 이 아닌 특정 사용자와 그룹만 기록하려는 경우 도 있습니다.
- **tlog_users_sssd:**
 - **recordeduser** 는 세션을 기록하려는 사용자를 지정합니다. 이 경우 사용자를 추가하지 않습니다. 사용자를 직접 설정해야 합니다.

2. 필요한 경우 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

결과적으로 플레이북은 지정한 시스템에 터미널 세션 기록 역할을 설치합니다. 또한 사용자가 정의한 사용자 및 그룹에서 사용할 수 있는 **SSSD** 구성 드롭 파일을 생성합니다. **SSSD**는 이러한 사용자와 그룹을 구문 분석하고 읽어 **tlog** 세션을 셸 사용자로 오버레이합니다. 또한 **cockpit** 패키지가 시스템에 설치된 경우 플레이북은 웹 콘솔 인터페이스에서 레코딩을 보고 재생할 수 있는 **Cockpit** 모듈인 **cockpit-session-Recording** 패키지도 설치합니다.

검증 단계

1. **/etc/journal.conf** 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 시스템이 로그에 메시지를 전송하는지 확인합니다.

SSSD 구성 드롭 파일이 시스템에서 생성되었는지 확인하려면 다음 단계를 수행하십시오.

1. **SSSD** 구성 드롭 파일이 생성된 폴더로 이동합니다.

```
# cd /etc/sssdcnf.d
```

2. 파일 내용을 확인합니다.

```
# cat sssd-session-recording.conf
```

파일에 플레이북에서 설정한 매개변수가 포함되어 있음을 확인할 수 있습니다.

5장. RHEL의 ANSIBLE IPMI 모듈

5.1. RHEL_BACKEND 컬렉션

IPMI(Intelligent Platform Management Interface)는 **BMC(Baseboard Management Controller)** 장치와 통신하는 표준 프로토콜 세트의 사양입니다. **IPMI** 모듈을 사용하면 하드웨어 관리 자동화를 활성화하고 지원할 수 있습니다. **IPMI** 모듈은 다음에서 사용할 수 있습니다.

- **rhel_backend** 컬렉션. 패키지 이름은 **ansible-collection-redhat-rhel_backend** 입니다.
- **RHEL 8 AppStream**은 새로운 **ansible-collection-redhat-rhel_retries** 패키지의 일부로 사용됩니다.

다음 **IPMI** 모듈은 **rhel_backend** 컬렉션에서 사용할 수 있습니다.

- **ipmi_boot**: 부팅 장치 순서 관리
- **ipmi_power**: 머신의 전원 관리

IPMI 모듈에 사용되는 필수 매개변수는 다음과 같습니다.

- **ipmi_boot** 매개 변수:

모듈 이름	설명
name	BMC의 호스트 이름 또는 IP 주소
암호	BMC에 연결할 암호

모듈 이름	설명
bootdev	다음 부팅 시 사용할 장치 * 네트워크 * 플로피 * HD * 안전 * 광장 * 설정 * 기본
사용자	BMC에 연결할 사용자 이름

- **ipmi_power parameters:**

모듈 이름	설명
name	BMC 호스트 이름 또는 IP 주소
암호	BMC에 연결할 암호
user	BMC에 연결할 사용자 이름
상태	시스템이 원하는 상태에 있는지 확인합니다. * On * 끄기 * 종료 * 초기화 * 부팅

5.2. CLI를 사용하여 RHEL RWX 컬렉션 설치

명령줄을 사용하여 **rhel_backend** 컬렉션을 설치할 수 있습니다.

사전 요구 사항

- **ansible-core** 패키지가 설치되어 있습니다.

절차

- **RPM** 패키지를 통해 컬렉션을 설치합니다.

```
# yum install ansible-collection-redhat-rhel_mgmt
```

설치가 완료되면 **redhat.rhel_ovs Ansible** 컬렉션에서 **IPMI** 모듈을 사용할 수 있습니다.

추가 리소스

- **ansible-playbook** 도움말 페이지.

5.3. IPMI_BOOT 모듈 사용 예

다음 예는 플레이북에서 **ipmi_boot** 모듈을 사용하여 다음 부팅에 대한 부팅 장치를 설정하는 방법을 보여줍니다. 간소화하기 위해 예제에서는 **Ansible** 제어 호스트 및 관리 호스트와 동일한 호스트를 사용하므로 플레이북이 실행되는 동일한 호스트에서 모듈을 실행합니다.

사전 요구 사항

- **rhel_backend** 컬렉션이 설치되어 있습니다.
- **python3-pyghmi** 패키지의 **pyghmi** 라이브러리는 다음 위치 중 하나에 설치됩니다.
 - 플레이북을 실행하는 호스트입니다.
 - 관리 호스트입니다. **localhost**를 관리 호스트로 사용하는 경우 플레이북을 대신 실행하는 호스트에 **python3-pyghmi** 패키지를 설치합니다.
- 제어하려는 **IPMI BMC**는 플레이북을 실행하는 호스트 또는 관리 호스트를 통해(관리 호스트로 사용하지 않는 경우) 네트워크를 통해 액세스할 수 있습니다. 모듈이 실행되고 있는 호스트(**Ansible** 관리 호스트)는 **IPMI** 프로토콜을 사용하여 네트워크를 통해 **BMC**에 연결하므로 일반적으로 모듈에서 **BMC**를 구성하는 호스트와는 다릅니다.

- 적절한 액세스 수준을 사용하여 **BMC**에 액세스할 수 있는 인증 정보가 있습니다.

절차

1. 다음 콘텐츠를 사용하여 새 **playbook.yml** 파일을 생성합니다.

```
---
- name: Sets which boot device will be used on next boot
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.ipmi_boot:
      name: bmc.host.example.com
      user: admin_user
      password: basics
      bootdev: hd
```

2. **localhost**에 대해 플레이북을 실행합니다.

```
# ansible-playbook playbook.yml
```

결과적으로 출력은 **"success"** 값을 반환합니다.

5.4. IPMI_POWER 모듈 사용 예

이 예는 플레이북에서 **ipmi_boot** 모듈을 사용하여 시스템이 설정되어 있는지 확인하는 방법을 보여줍니다. 간소화하기 위해 예제에서는 **Ansible** 제어 호스트 및 관리 호스트와 동일한 호스트를 사용하므로 플레이북이 실행되는 동일한 호스트에서 모듈을 실행합니다.

사전 요구 사항

- **rhel_backend** 컬렉션이 설치되어 있습니다.
- **python3-pyghmi** 패키지의 **pyghmi** 라이브러리는 다음 위치 중 하나에 설치됩니다.
 - 플레이북을 실행하는 호스트입니다.
 - 관리 호스트입니다. **localhost**를 관리 호스트로 사용하는 경우 플레이북을 대신 실행하

는 호스트에 **python3-pyghmi** 패키지를 설치합니다.

- 제어하려는 **IPMI BMC**는 플레이북을 실행하는 호스트 또는 관리 호스트를 통해(관리 호스트로 사용하지 않는 경우) 네트워크를 통해 액세스할 수 있습니다. 모듈이 실행되고 있는 호스트 (**Ansible** 관리 호스트)는 **IPMI** 프로토콜을 사용하여 네트워크를 통해 **BMC**에 연결하므로 일반적으로 모듈에서 **BMC**를 구성하는 호스트와는 다릅니다.
- 적절한 액세스 수준을 사용하여 **BMC**에 액세스할 수 있는 인증 정보가 있습니다.

절차

1.

다음 콘텐츠를 사용하여 새 **playbook.yml** 파일을 생성합니다.

```
---
- name: Turn the host on
  hosts: localhost
  tasks:
    - redhat.rhel_mgmt.ipmi_power:
      name: bmc.host.example.com
      user: admin_user
      password: basics
      state: on
```

2.

Playbook을 실행합니다.

```
# ansible-playbook playbook.yml
```

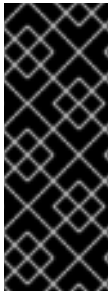
출력은 **"true"** 값을 반환합니다.

6장. ANSIBLE 역할을 사용하여 커널 매개 변수를 영구적으로 구성

커널 설정 역할을 사용하여 한 번에 여러 클라이언트에서 커널 매개변수를 구성할 수 있습니다. 이 해결 방법:

- 효율적인 입력 설정을 통해 친숙한 인터페이스를 제공합니다.
- 의도한 모든 커널 매개변수를 한 곳에 유지합니다.

제어 머신에서 커널 설정 역할을 실행하면 커널 매개변수가 관리 시스템에 즉시 적용되고 재부팅 시 지속됩니다.



중요

RHEL 채널을 통해 제공되는 RHEL 시스템 역할은 기본 **AppStream** 리포지토리의 RPM 패키지로 RHEL 고객이 사용할 수 있습니다. RHEL 시스템 역할은 **Ansible Automation Hub**를 통해 **Ansible** 서브스크립션을 사용하는 고객에 대한 컬렉션으로도 사용할 수 있습니다.

6.1. 커널 설정 역할 소개

RHEL 시스템 역할은 여러 시스템을 원격으로 관리할 수 있도록 일관된 구성 인터페이스를 제공하는 역할 집합입니다.

RHEL 시스템 역할은 커널 설정 시스템 역할을 사용하여 커널 자동 구성을 위해 도입되었습니다. **rhel-system-roles** 패키지에는 이 시스템 역할과 참조 문서도 포함됩니다.

자동화된 방식으로 하나 이상의 시스템에 커널 매개변수를 적용하려면 플레이북에서 선택한 역할 변수 중 하나 이상과 함께 커널 설정 역할을 사용합니다. 플레이북은 사람이 읽을 수 있고 **YAML** 형식으로 작성된 하나 이상의 플레이 목록입니다.

커널 설정 역할을 사용하여 다음을 구성할 수 있습니다.

- **kernel_settings_sysctl** 역할 변수를 사용하는 커널 매개변수

- **kernel_settings_sysfs** 역할 변수를 사용하는 다양한 커널 하위 시스템, 하드웨어 장치 및 장치 드라이버
- **kernel_settings_systemd_cpu_affinity** 역할 변수를 사용하여 **systemd** 서비스 관리자 및 프로세스용 CPU 선호도
- 커널 메모리 하위 시스템은 **kernel_settings_transparent_hugepages** 및 **kernel_settings_transparent_hugepages_defrag** 역할 변수를 사용하여 **hugepages**를 투명하게 합니다.

추가 리소스

- **README.md** 및 **README.html** 파일(`/usr/share/doc/rhel-system-roles/kernel_settings/directory`)
- [플레이북 작업](#)
- [인벤토리를 빌드하는 방법](#)

6.2. 커널 설정 역할을 사용하여 선택한 커널 매개변수 적용

다음 단계에 따라 여러 관리 운영 체제에 영향을 주므로 **Ansible** 플레이북을 준비하고 적용하여 커널 매개 변수를 원격으로 구성합니다.

사전 요구 사항

- **root** 권한이 있습니다.
- **RHEL** 서브스크립션에서 권한을 부여하면 컨트롤 시스템에 **ansible-core** 및 **rhel-system-roles** 패키지를 설치했습니다.
- 관리 호스트의 인벤토리는 제어 시스템에 있으며 **Ansible**은 연결할 수 있습니다.

중요

RHEL 8.0 - 8.5에서는 Ansible을 기반으로 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스 권한을 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 플러그인 및 모듈 전체와 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [How do I download and Install Red Hat Ansible Engine](#) 을 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible Core(Ansible 명령줄 유틸리티, 명령 및 소규모 Ansible 플러그인 세트가 포함된 Ansible 코어 RPM 제공)를 도입했습니다. AppStream 리포지토리는 제한된 지원 범위가 있는 **ansible-core** 를 제공합니다. [RHEL 9 AppStream](#) 에 포함된 **ansible-core** 패키지에 대한 지원 범위를 검토하여 자세한 내용을 확인할 수 있습니다.

절차

1.

선택적으로 설명을 위해 인벤토리 파일을 검토합니다.

```
# cat /home/jdoe/<ansible_project_name>/inventory
[testingservers]
pdoe@192.168.122.98
fdoe@192.168.122.226

[db-servers]
db1.example.com
db2.example.com

[webservers]
web1.example.com
web2.example.com
192.0.2.42
```

파일은 **[testingservers]** 그룹 및 기타 그룹을 정의합니다. 이를 통해 특정 시스템 집합에 대해 Ansible을 더 효과적으로 실행할 수 있습니다.

2.

구성 파일을 생성하여 Ansible 작업에 대한 기본값 및 권한 에스컬레이션을 설정합니다.

a.

새 YAML 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

```
# vi /home/jdoe/<ansible_project_name>/ansible.cfg
```

b.

파일에 다음 내용을 삽입합니다.

```
[defaults]
inventory = ./inventory

[privilege_escalation]
become = true
become_method = sudo
become_user = root
become_ask_pass = true
```

[defaults] 섹션은 관리 호스트의 인벤토리 파일의 경로를 지정합니다.

[privilege_escalation] 섹션은 사용자 권한을 지정된 관리 호스트에서 **root** 로 변경하는 것을 정의합니다. 이는 커널 매개 변수를 성공적으로 구성하는 데 필요합니다. **Ansible** 플레이북이 실행되면 사용자 암호를 입력하라는 메시지가 표시됩니다. 사용자는 관리 호스트에 연결한 후 **sudo** 를 통해 **root** 로 자동 전환합니다.

3.

커널 설정 역할을 사용하는 **Ansible** 플레이북을 생성합니다.

a.

새 **YAML** 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

```
# vi /home/jdoe/<ansible_project_name>/kernel-roles.yml
```

이 파일은 플레이북을 나타내며 일반적으로 인벤토리 파일에서 선택한 특정 관리 호스트에 대해 실행되는 **플레이** 라고도 하는 순서가 지정된 작업 목록을 포함합니다.

b.

파일에 다음 내용을 삽입합니다.

```
---
-
  hosts: testingservers
  name: "Configure kernel settings"
  roles:
    - rhel-system-roles.kernel_settings
  vars:
    kernel_settings_sysctl:
      - name: fs.file-max
        value: 400000
      - name: kernel.threads-max
        value: 65536
    kernel_settings_sysfs:
      - name: /sys/class/net/lo/mtu
        value: 65000
    kernel_settings_transparent_hugepages: madvise
```

name 키는 선택 사항입니다. 임의의 문자열을 플레이와 레이블로 연결하고 플레이의 기능을 식별합니다. 플레이의 **hosts** 키는 플레이가 실행되는 호스트를 지정합니다. 이 키의 값 또는 값은 관리 호스트의 개별 이름 또는 인벤토리 파일에 정의된 호스트 그룹으로 제공될 수 있습니다.

vars 섹션은 선택한 커널 매개변수 이름과 값을 설정해야 하는 변수 목록을 나타냅니다.

roles 키는 **vars** 섹션에 언급된 매개변수와 값을 구성하려는 시스템 역할을 지정합니다.



참고

필요에 맞게 플레이북에서 커널 매개변수 및 해당 값을 수정할 수 있습니다.

4.

선택적으로 플레이의 구문이 올바른지 확인합니다.

```
# ansible-playbook --syntax-check kernel-roles.yml
```

```
playbook: kernel-roles.yml
```

이 예는 플레이북의 성공적인 확인을 보여줍니다.

5.

플레이북을 실행합니다.

```
# ansible-playbook kernel-roles.yml
```

```
...
```

```
BECOME password:
```

```
PLAY [Configure kernel settings]
```

```
*****
```

```
PLAY RECAP
```

```
*****
```

```
fdoe@192.168.122.226 : ok=10 changed=4 unreachable=0 failed=0 skipped=6
rescued=0 ignored=0
pdoe@192.168.122.98 : ok=10 changed=4 unreachable=0 failed=0 skipped=6
rescued=0 ignored=0
```

Ansible이 플레이북을 실행하기 전에 암호를 입력하라는 메시지가 표시되고 관리 호스트의 사용자가 커널 매개 변수를 구성하는 데 필요한 **root** 으로 전환할 수 있습니다.

recap 섹션은 모든 관리 호스트에 대해 플레이가 성공적으로 완료됨(**failed=0**)하고 4 커널 매개 변수가 적용됨을 보여줍니다(**changed=4**).

6.

관리 호스트를 재시작하고 영향을 받는 커널 매개 변수를 확인하여 변경 사항이 적용되고 재부팅해도 지속되는지 확인합니다.

추가 리소스

- [RHEL 시스템 역할 시작하기](#)
- [README.html 및 README.md 파일 /usr/share/doc/rhel-system-roles/kernel_settings/ 디렉터리에](#)
- [인벤토리 빌드](#)
- [Ansible 구성](#)
- [플레이북으로 작업](#)
- [변수 사용](#)
- [역할](#)

7장. RHEL 시스템 역할을 사용하여 네트워크 연결 구성

관리자는 네트워킹 RHEL 시스템 역할을 통해 **Ansible**을 사용하여 네트워크 관련 구성 및 관리 작업을 자동화할 수 있습니다.

7.1. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 다음 설정으로 **enp7s0** 인터페이스에 대한 이더넷 연결을 원격으로 추가하는 데 네트워킹 RHEL 시스템 역할을 사용하는 방법을 설명합니다.

- 정적 IPv4 주소 - 192.0.2.1 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 192.0.2.254
- IPv6 기본 게이트웨이 - 2001:db8:1::fffe
- IPv4 DNS 서버 - 192.0.2.200
- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com

Ansible 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-static-IP.yml** 플레이북을 생성합니다.

```
---
- name: Configure an Ethernet connection with static IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
      name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp7s0
        interface_name: enp7s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 192.0.2.1/24
            - 2001:db8:1::1/64
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
        dns:
          - 192.0.2.200
          - 2001:db8:1::ffbb
        dns_search:
          - example.com
        state: up
```

3. 플레이북을 실행합니다.

- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-static-IP.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)
- [ansible-playbook\(1\) 도움말 페이지](#)

7.2. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 특정 장치 경로와 일치하는 장치의 고정 **IP** 주소와 함께 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다.

다음 명령을 사용하여 장치 경로를 식별할 수 있습니다.

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

이 절차에서는 **PCI ID 0000:00:0[1-3].0** 표현식과 일치하는 장치로 다음 설정을 설정하지만 **0000:00:02.0** 은 아닙니다.

- 정적 **IPv4** 주소 - **192.0.2.1** 에 /24 서브넷 마스크
- 정적 **IPv6** 주소 - **2001:db8:1::1** 에 /64 서브넷 마스크
- **IPv4** 기본 게이트웨이 - **192.0.2.254**

- **IPv6 기본 게이트웨이 - 2001:db8:1::fffe**
- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**

Ansible 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-dynamic-IP.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
      name: rhel-system-roles.network
```

```
vars:
  network_connections:
    - name: example
      match:
        path:
          - pci-0000:00:0[1-3].0
          - &!pci-0000:00:02.0
        type: ethernet
        autoconnect: yes
      ip:
        address:
          - 192.0.2.1/24
          - 2001:db8:1::1/64
        gateway4: 192.0.2.254
        gateway6: 2001:db8:1::fffe
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
      state: up
```

이 예제의 `match` 매개 변수는 Ansible이 PCI ID `0000:00:0[1-3].0` 과 일치하는 장치에 플레이를 적용하지만 `0000:00:00:02.0`이 아닌 장치에 플레이를 적용하도록 정의합니다. 사용할 수 있는 특수 수정자 및 와일드카드에 대한 자세한 내용은 `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 파일에서 일치하는 매개 변수 설명을 참조하십시오.

3.

플레이북을 실행합니다.

-

`root` 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

`--ask-become-pass` 옵션을 사용하면 `ansible-playbook` 명령에서 `-u user_name` 옵션에 정의된 사용자의 `sudo` 암호를 입력하라는 메시지를 표시합니다.

`-u user_name` 옵션을 지정하지 않으면 `ansible-playbook` 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` file
- [ansible-playbook\(1\)](#) 도움말 페이지

7.3. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 **enp7s0** 인터페이스에 대해 동적 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다. 이 설정을 사용하면 네트워크 연결에서 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다. **Ansible** 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- **DHCP** 서버는 네트워크에서 사용할 수 있습니다.
- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 `/etc/ansible/hosts` **Ansible** 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 `~/ethernet-dynamic-IP.yml` 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
  - include_role:
```

```

name: rhel-system-roles.network

vars:
  network_connections:
    - name: enp7s0
      interface_name: enp7s0
      type: ethernet
      autoconnect: yes
      ip:
        dhcp4: yes
        auto6: yes
      state: up

```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

•

[/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file

•

[ansible-playbook\(1\)](#) 도움말 페이지

7.4. 장치 경로가 있는 RHEL 시스템 역할을 사용하여 동적 이더넷 연결 구성

이 절차에서는 **Ansible** 플레이북을 실행하여 특정 장치 경로와 일치하는 장치에 대한 동적 이더넷 연결을 원격으로 추가하는 데 **RHEL** 시스템 역할을 사용하는 방법을 설명합니다. 동적 IP 설정을 사용하면 네

트위크 연결이 **DHCP** 서버에서 이 연결에 대한 **IP** 설정을 요청합니다. **Ansible** 제어 노드에서 다음 절차를 실행합니다.

다음 명령을 사용하여 장치 경로를 식별할 수 있습니다.

```
# udevadm info /sys/class/net/<device_name> | grep ID_PATH=
```

사전 요구 사항

- **DHCP** 서버는 네트워크에서 사용할 수 있습니다.
- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 호스트는 **NetworkManager**를 사용하여 네트워크를 구성합니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/ethernet-dynamic-IP.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with dynamic IP
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: example
        match:
          path:
```

```

- pci-0000:00:0[1-3].0
- &!pci-0000:00:02.0
type: ethernet
autoconnect: yes
ip:
  dhcp4: yes
  auto6: yes
state: up

```

이 예제의 **match** 매개 변수는 Ansible이 PCI ID 0000:00:0[1-3].0 과 일치하는 장치에 플레이를 적용하지만 0000:00:00 :02.0이 아닌 장치에 플레이를 적용하도록 정의합니다. 사용할 수 있는 특수 수정자 및 와일드카드에 대한 자세한 내용은 `/usr/share/ansible/roles/rhel-system-roles.network/README.md` 파일에서 일치하는 매개변수 설명을 참조하십시오.

3.

플레이북을 실행합니다.

- **root** 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-dynamic-IP.yml
```

- 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-dynamic-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.network/README.md` file
- **ansible-playbook(1)** 도움말 페이지

7.5. RHEL 시스템 역할을 사용하여 VLAN 태그 지정 구성

Networking RHEL 시스템 역할을 사용하여 **VLAN** 태그를 구성할 수 있습니다. 이 절차에서는 이 이더넷 연결 상단에 **ID가 10인 VLAN**과 이더넷 연결을 추가하는 방법을 설명합니다. 하위 장치로 **VLAN** 연결에는 **IP**, 기본 게이트웨이 및 **DNS** 구성이 포함됩니다.

환경에 따라 적절하게 플레이를 조정합니다. 예를 들면 다음과 같습니다.

- 본딩과 같은 다른 연결의 포트로 **VLAN**을 사용하려면 **ip** 속성을 생략하고 하위 구성에서 **IP** 구성을 설정합니다.
- **VLAN**에서 팀, 브리지 또는 본딩 장치를 사용하려면 **VLAN**에서 사용하는 포트의 **interface_name** 및 **type** 속성을 조정합니다.

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/vlan-ethernet.yml** 플레이북을 생성합니다.

```
---
- name: Configure a VLAN that uses an Ethernet connection
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      # Add an Ethernet profile for the underlying device of the VLAN
      - name: enp1s0
```

```

type: ethernet
interface_name: enp1s0
autoconnect: yes
state: up
ip:
  dhcp4: no
  auto6: no

# Define the VLAN profile
- name: enp1s0.10
  type: vlan
  ip:
    address:
      - "192.0.2.1/24"
      - "2001:db8:1::1/64"
    gateway4: 192.0.2.254
    gateway6: 2001:db8:1::fffe
  dns:
    - 192.0.2.200
    - 2001:db8:1::ffbb
  dns_search:
    - example.com
  vlan_id: 10
  parent: enp1s0
  state: up

```

VLAN 프로파일의 상위 속성은 **enp1s0** 장치 상단에서 작동하도록 VLAN을 구성합니다.

3.

플레이북을 실행합니다.

-

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/vlan-ethernet.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/vlan-ethernet.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file
- [ansible-playbook\(1\)](#) 도움말 페이지

7.6. RHEL 시스템 역할을 사용하여 네트워크 브리지 구성

네트워킹 **RHEL** 시스템 역할을 사용하여 **Linux** 브리지를 구성할 수 있습니다. 이 절차에서는 두 개의 이더넷 장치를 사용하는 네트워크 브릿지를 구성하고 **IPv4** 및 **IPv6** 주소, 기본 게이트웨이 및 **DNS** 구성을 설정하는 방법을 설명합니다.



참고

Linux 브리지 포트에 없는 브릿지에 **IP** 구성을 설정합니다.

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 서버에 두 개 이상의 물리적 또는 가상 네트워크 장치가 설치되어 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/bridge-ethernet.yml** 플레이북을 만듭니다.

```
---
- name: Configure a network bridge that uses two Ethernet ports
```

```

hosts: node.example.com
become: true
tasks:
- include_role:
    name: rhel-system-roles.network

vars:
  network_connections:
    # Define the bridge profile
    - name: bridge0
      type: bridge
      interface_name: bridge0
      ip:
        address:
          - "192.0.2.1/24"
          - "2001:db8:1::1/64"
        gateway4: 192.0.2.254
        gateway6: 2001:db8:1::fffe
      dns:
        - 192.0.2.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
      state: up

    # Add an Ethernet profile to the bridge
    - name: bridge0-port1
      interface_name: enp7s0
      type: ethernet
      controller: bridge0
      port_type: bridge
      state: up

    # Add a second Ethernet profile to the bridge
    - name: bridge0-port2
      interface_name: enp8s0
      type: ethernet
      controller: bridge0
      port_type: bridge
      state: up

```

3.

플레이북을 실행합니다.

-

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/bridge-ethernet.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/bridge-ethernet.yml
```

■

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md file](#)
- [ansible-playbook\(1\) 도움말 페이지](#)

7.7. RHEL 시스템 역할을 사용하여 네트워크 본딩 구성

Networking RHEL 시스템 역할을 사용하여 네트워크 본딩을 구성할 수 있습니다. 이 절차에서는 두 개의 이더넷 장치를 사용하는 **active-backup** 모드로 본딩을 구성하는 방법을 설명하고 **IPv4** 및 **IPv6** 주소, 기본 게이트웨이 및 **DNS** 구성을 설정합니다.



참고

Linux 본딩 포트에가 아니라 본딩에서 **IP** 구성을 설정합니다.

사전 요구 사항

- **ansible-core** 패키지 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.
- 서버에 두 개 이상의 물리적 또는 가상 네트워크 장치가 설치되어 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

node.example.com

2.

다음 콘텐츠를 사용하여 **~/bond-ethernet.yml** 플레이북을 생성합니다.

```
---
- name: Configure a network bond that uses two Ethernet ports
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      # Define the bond profile
      - name: bond0
        type: bond
        interface_name: bond0
        ip:
          address:
            - "192.0.2.1/24"
            - "2001:db8:1::1/64"
          gateway4: 192.0.2.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 192.0.2.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        bond:
          mode: active-backup
          state: up

      # Add an Ethernet profile to the bond
      - name: bond0-port1
        interface_name: enp7s0
        type: ethernet
        controller: bond0
        state: up

      # Add a second Ethernet profile to the bond
      - name: bond0-port2
        interface_name: enp8s0
        type: ethernet
        controller: bond0
        state: up
```

3.

플레이북을 실행합니다.

●

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/bond-ethernet.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/bond-ethernet.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

-

[/usr/share/ansible/roles/rhel-system-roles.network/README.md file](#)

-

[ansible-playbook\(1\) 도움말 페이지](#)

7.8. RHEL 시스템 역할을 사용하여 802.1X 네트워크 인증으로 정적 이더넷 연결 구성

네트워킹 **RHEL** 시스템 역할을 사용하면 **802.1X** 표준을 사용하여 클라이언트를 인증하는 이더넷 연결 생성을 자동화할 수 있습니다. 이 절차에서는 **Ansible** 플레이북을 실행하여 다음 설정으로 **enp1s0** 인터페이스에 대한 이더넷 연결을 원격으로 추가하는 방법을 설명합니다.

-

정적 **IPv4** 주소 - **192.0.2.1** 에 **/24** 서브넷 마스크

-

정적 **IPv6** 주소 - **2001:db8:1::1** 에 **/64** 서브넷 마스크

-

IPv4 기본 게이트웨이 - **192.0.2.254**

-

IPv6 기본 게이트웨이 - **2001:db8:1::fffe**

- **IPv4 DNS 서버 - 192.0.2.200**
- **IPv6 DNS 서버 - 2001:db8:1::ffbb**
- **DNS 검색 도메인 - example.com**
- **TLS EAP(Extensible Authentication Protocol)를 사용한 802.1x 네트워크 인증**

Ansible 제어 노드에서 다음 절차를 실행합니다.

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 관리 노드에 대한 적절한 **sudo** 권한이 있어야 합니다.
- 네트워크는 **802.1X** 네트워크 인증을 지원합니다.
- 관리형 노드는 **NetworkManager**를 사용합니다.
- **TLS** 인증에 필요한 다음 파일이 제어 노드에 있습니다.
 - 클라이언트 키는 **/srv/data/client.key** 파일에 저장됩니다.
 - 클라이언트 인증서는 **/srv/data/client.crt** 파일에 저장됩니다.
 - **CA(인증 기관)** 인증서는 **/srv/data/ca.crt** 파일에 저장됩니다.

절차

1.

플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2.

다음 콘텐츠를 사용하여 **~/enable-802.1x.yml** 플레이북을 생성합니다.

```
---
- name: Configure an Ethernet connection with 802.1X authentication
  hosts: node.example.com
  become: true
  tasks:
    - name: Copy client key for 802.1X authentication
      copy:
        src: "/srv/data/client.key"
        dest: "/etc/pki/tls/private/client.key"
        mode: 0600

    - name: Copy client certificate for 802.1X authentication
      copy:
        src: "/srv/data/client.crt"
        dest: "/etc/pki/tls/certs/client.crt"

    - name: Copy CA certificate for 802.1X authentication
      copy:
        src: "/srv/data/ca.crt"
        dest: "/etc/pki/ca-trust/source/anchors/ca.crt"

  - include_role:
      name: rhel-system-roles.network
    vars:
      network_connections:
        - name: enp1s0
          type: ethernet
          autoconnect: yes
          ip:
            address:
              - 192.0.2.1/24
              - 2001:db8:1::1/64
            gateway4: 192.0.2.254
            gateway6: 2001:db8:1::fffe
            dns:
              - 192.0.2.200
              - 2001:db8:1::ffbb
            dns_search:
              - example.com
      ieee802_1x:
        identity: user_name
        eap: tls
        private_key: "/etc/pki/tls/private/client.key"
        private_key_password: "password"
        client_cert: "/etc/pki/tls/certs/client.crt"
```

```
ca_cert: "/etc/pki/ca-trust/source/anchors/ca.crt"
domain_suffix_match: example.com
state: up
```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/enable-802.1x.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-static-IP.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

•

/usr/share/ansible/roles/rhel-system-roles.network/README.md file

•

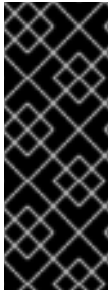
/usr/share/ansible/roles/rhel-system-roles.network/README.md file

•

ansible-playbook(1) 도움말 페이지

7.9. SYSTEM ROLES를 사용하여 기존 연결에 기본 게이트웨이 설정

Networking RHEL 시스템 역할을 사용하여 기본 게이트웨이를 설정할 수 있습니다.



중요

Networking RHEL 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 것과 일치하지 않는 경우 시스템 역할은 이름이 동일한 기존 연결 프로필을 덮어씁니다. 따라서 **IP** 구성이 이미 있는 경우에도 항상 플레이에서 네트워크 연결 프로필의 전체 구성을 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 **IPv4** 주소 - **198.51.100.20** 에 /24 서브넷 마스크
- 정적 **IPv6** 주소 - **2001:db8:1::1** 에 /64 서브넷 마스크
- **IPv4** 기본 게이트웨이 - **198.51.100.254**
- **IPv6** 기본 게이트웨이 - **2001:db8:1::fffe**
- **IPv4** DNS 서버 **198.51.100.200**
- **IPv6** DNS 서버 - **2001:db8:1::ffbb**
- DNS 검색 도메인 - **example.com**

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

절차

- 1.

플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2.

다음 콘텐츠를 사용하여 **~/ethernet-connection.yml** 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with static IP and default gateway
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp1s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 198.51.100.20/24
            - 2001:db8:1::1/64
          gateway4: 198.51.100.254
          gateway6: 2001:db8:1::fffe
          dns:
            - 198.51.100.200
            - 2001:db8:1::ffbb
          dns_search:
            - example.com
        state: up
```

3.

플레이북을 실행합니다.

-

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/ethernet-connection.yml
```

-

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/ethernet-connection.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

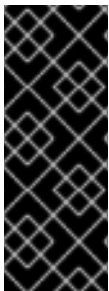
-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)
- [ansible-playbook\(1\)](#) 도움말 페이지

7.10. RHEL 시스템 역할을 사용하여 정적 경로 구성

Networking RHEL 시스템 역할을 사용하여 정적 경로를 구성할 수 있습니다.



중요

Networking RHEL 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 것과 일치하지 않는 경우 시스템 역할은 이름이 동일한 기존 연결 프로필을 덮어씁니다. 따라서 IP 구성이 이미 있는 경우에도 항상 플레이에서 네트워크 연결 프로필의 전체 구성을 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp7s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - 198.51.100.20 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 198.51.100.254
- IPv6 기본 게이트웨이 - 2001:db8:1::fffe

- IPv4 DNS 서버 198.51.100.200
- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com
- 정적 경로:
 - 192.0.2.0/24 게이트웨이 198.51.100.1
 - 203.0.113.0/24 게이트웨이 198.51.100.2

사전 요구 사항

- **ansible-core** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 **~/add-static-routes.yml** 플레이북을 생성합니다.

```
---
- name: Configure an Ethernet connection with static IP and additional routes
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network
```

```
vars:
  network_connections:
    - name: enp7s0
      type: ethernet
      autoconnect: yes
      ip:
        address:
          - 198.51.100.20/24
          - 2001:db8:1::1/64
        gateway4: 198.51.100.254
        gateway6: 2001:db8:1::fffe
      dns:
        - 198.51.100.200
        - 2001:db8:1::ffbb
      dns_search:
        - example.com
      route:
        - network: 192.0.2.0
          prefix: 24
          gateway: 198.51.100.1
        - network: 203.0.113.0
          prefix: 24
          gateway: 198.51.100.2
      state: up
```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/add-static-routes.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/add-static-routes.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

검증 단계

- 라우팅 테이블을 표시합니다.

```
# ip -4 route
default via 198.51.100.254 dev enp7s0 proto static metric 100
192.0.2.0/24 via 198.51.100.1 dev enp7s0 proto static metric 100
203.0.113.0/24 via 198.51.100.2 dev enp7s0 proto static metric 100
...
```

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md](#) file
- [ansible-playbook\(1\)](#) 도움말 페이지

7.11. RHEL 시스템 역할을 사용하여 ETHTOOL 기능 설정

Networking RHEL 시스템 역할을 사용하여 NetworkManager 연결의 **ethtool** 기능을 구성할 수 있습니다.



중요

Networking RHEL 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 것과 일치하지 않는 경우 시스템 역할은 이름이 동일한 기존 연결 프로필을 덮어씁니다. 따라서 IP 구성과 같이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할이 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - 198.51.100.20 에 /24 서브넷 마스크
- 정적 IPv6 주소 - 2001:db8:1::1 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - 198.51.100.254

- IPv6 기본 게이트웨이 - 2001:db8:1::fffe
- IPv4 DNS 서버 198.51.100.200
- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com
- ethtool 기능:
 - 일반 수신 오프로드 (GRO): 비활성화
 - 범용 분할 오프로드(GSO): 활성화됨
 - TX 스트림 제어 전송 프로토콜 (SCTP) 세그멘테이션:

사전 요구 사항

- **ansible-core** 패키지 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root**와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 **IP** 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.
 **node.example.com**
2. 다음 콘텐츠를 사용하여 **~/configure-ethernet-device-with-ethtool-features.yml** 플레이북을 만듭니다.

```

---
- name: Configure an Ethernet connection with ethtool features
  hosts: node.example.com
  become: true
  tasks:
  - include_role:
      name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp1s0
        type: ethernet
        autoconnect: yes
        ip:
          address:
            - 198.51.100.20/24
            - 2001:db8:1::1/64
          gateway4: 198.51.100.254
          gateway6: 2001:db8:1::fffe
        dns:
          - 198.51.100.200
          - 2001:db8:1::ffbb
        dns_search:
          - example.com
        ethtool:
          features:
            gro: "no"
            gso: "yes"
            tx_sctp_segmentation: "no"
        state: up

```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/configure-ethernet-device-with-ethtool-features.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-with-ethtool-features.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u user_name** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u user_name 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사

용자로 관리 호스트에 연결합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.network/README.md file](#)
- [ansible-playbook\(1\) 도움말 페이지](#)

7.12. RHEL 시스템 역할을 사용하여 ETHTOOL COALESCE 설정 설정

Networking RHEL 시스템 역할을 사용하여 **NetworkManager** 연결의 **ethtool BIOSesce** 설정을 구성할 수 있습니다.



중요

Networking RHEL 시스템 역할을 사용하는 플레이를 실행하면 설정 값이 플레이에 지정된 것과 일치하지 않는 경우 시스템 역할은 이름이 동일한 기존 연결 프로필을 덮어씁니다. 따라서 IP 구성과 같이 이미 존재하는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할이 이러한 값을 기본값으로 재설정합니다.

이 절차가 이미 존재하는지 여부에 따라 다음 설정으로 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20** 에 /24 서브넷 마스크
- 정적 IPv6 주소 - **2001:db8:1::1** 에 /64 서브넷 마스크
- IPv4 기본 게이트웨이 - **198.51.100.254**
- IPv6 기본 게이트웨이 - **2001:db8:1::fffe**
- IPv4 DNS 서버 **198.51.100.200**

- IPv6 DNS 서버 - 2001:db8:1::ffbb
- DNS 검색 도메인 - example.com
- ethtool Coesce 설정:
 - RX 프레임: 128
 - TX 프레임: 128

사전 요구 사항

- ansible-core 및 rhel-system-roles 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 root와 다른 원격 사용자를 사용하는 경우 이 사용자는 관리 노드에 대한 적절한 sudo 권한이 있습니다.

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 생성되지 않은 경우 이 호스트의 IP 또는 이름을 /etc/ansible/hosts Ansible 인벤토리 파일에 추가합니다.

```
node.example.com
```

2. 다음 콘텐츠를 사용하여 ~/configure-ethernet-device-with-ethtoolcoalesce-settings.yml 플레이북을 만듭니다.

```
---
- name: Configure an Ethernet connection with ethtool coalesce settings
  hosts: node.example.com
  become: true
  tasks:
    - include_role:
        name: rhel-system-roles.network

  vars:
    network_connections:
      - name: enp1s0
```

```

type: ethernet
autoconnect: yes
ip:
  address:
    - 198.51.100.20/24
    - 2001:db8:1::1/64
  gateway4: 198.51.100.254
  gateway6: 2001:db8:1::fffe
  dns:
    - 198.51.100.200
    - 2001:db8:1::ffbb
  dns_search:
    - example.com
ethtool:
  coalesce:
    rx_frames: 128
    tx_frames: 128
state: up

```

3.

플레이북을 실행합니다.

•

root 사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u root ~/configure-ethernet-device-with-ethtoolcoalesce-
settings.yml
```

•

사용자로 관리 호스트에 연결하려면 다음을 입력합니다.

```
# ansible-playbook -u user_name --ask-become-pass ~/configure-ethernet-device-
with-ethtoolcoalesce-settings.yml
```

--ask-become-pass 옵션을 사용하면 **ansible-playbook** 명령에서 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

-u *user_name* 옵션을 지정하지 않으면 **ansible-playbook** 이 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

추가 리소스

•

[/usr/share/ansible/roles/rhel-system-roles.network/README.md](#)

- **ansible-playbook(1) 도움말 페이지**

8장. 시스템 역할의 **POSTFIX** 역할 변수

Postfix 역할 변수를 사용하면 사용자가 **Postfix** 메일 전송 에이전트(MTA)를 설치, 구성 및 시작할 수 있습니다.

다음 역할 변수는 이 섹션에 정의됩니다.

- **postfix_conf**: 지원되는 모든 **Postfix** 구성 매개 변수의 키/값 쌍을 포함합니다. 기본적으로 **postfix_conf**에는 값이 없습니다.

For example: ``postfix_conf`:`
`relayhost: "example.com"`

- **postfix_check**: **Postfix**를 시작하기 전에 검사가 실행되었는지 확인하여 구성 변경 사항을 확인합니다. 기본값은 **true**입니다.

For example: ``postfix_check: true``

- **postfix_backup**: 구성의 단일 백업 복사본이 생성되는지 여부를 확인합니다. 기본적으로 **postfix_backup** 값은 **false**입니다.

이전 백업을 덮어쓰려면 다음 명령을 실행합니다.

```
cp /etc/postfix/main.cf /etc/postfix/main.cf.backup
```

postfix_backup 값이 **true**로 변경되는 경우 **postfix_backup_multiple** 값도 **false**로 설정해야 합니다.

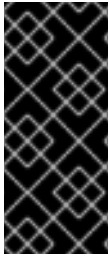
For example: `postfix_backup: true`
`postfix_backup_multiple: false`

- **postfix_backup_multiple**: 역할이 구성의 타임스탬프가 백업된 백업 사본을 만들지 여부를 결정합니다.

여러 백업 사본을 유지하려면 다음 명령을 실행합니다.

```
cp /etc/postfix/main.cf /etc/postfix/main.cf.$(date -lsec)
```

기본적으로 `postfix_backup_multiple` 값은 `true`입니다. `postfix_backup_multiple:true` 설정은 `postfix_backup` 을 덮어씁니다. `postfix_backup`을 사용하려면 `postfix_backup_multiple:false` 를 설정해야 합니다.



중요

구성 매개 변수는 제거할 수 없습니다. **Postfix** 역할을 실행하기 전에 `postfix_conf` 를 모든 필수 구성 매개 변수로 설정하고 `file` 모듈을 사용하여 `/etc/postfix/main.cf`를 제거합니다.

8.1. 추가 리소스

- </usr/share/doc/rhel-system-roles/postfix/README.md>

9장. 시스템 역할을 사용하여 SELINUX 구성

9.1. SELINUX 시스템 역할 소개

RHEL 시스템 역할은 여러 **RHEL** 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 **Ansible** 역할 및 모듈의 컬렉션입니다. **SELinux** 시스템 역할을 사용하면 다음 작업을 수행할 수 있습니다.

- **SELinux** 부울, 파일 컨텍스트, 포트 및 로그인과 관련된 로컬 정책 수정을 정리합니다.
- **SELinux** 정책 부울 설정, 파일 컨텍스트, 포트, 로그인.
- 지정된 파일 또는 디렉터리에서 파일 컨텍스트 복원.
- **SELinux** 모듈 관리.

다음 표에서는 **SELinux** 시스템 역할에서 사용할 수 있는 입력 변수에 대한 개요를 제공합니다.

표 9.1. SELinux 시스템 역할 변수

역할 변수	설명	CLI 대체
selinux_policy	대상 프로세스 또는 다중 수준 보안 보호를 보호하는 정책을 선택합니다.	/etc/selinux/config 의 SELINUXTYPE
selinux_state	은 SELinux 모드를 대체합니다.	/etc/selinux/config 의 setenforce 및 SELINUX .
selinux_booleans	SELinux 부울을 활성화 및 비활성화합니다.	setsebool
selinux_fcontexts	SELinux 파일 컨텍스트 매핑을 추가하거나 제거합니다.	semanage fcontext
selinux_restore_dirs	파일 시스템 트리에서 SELinux 레이블을 복원합니다.	restorecon -R
selinux_ports	포트에 SELinux 레이블을 설정합니다.	semanage port

역할 변수	설명	CLI 대체
selinux_logins	사용자를 SELinux 사용자 매핑으로 설정합니다.	semanage login
selinux_modules	SELinux 모듈을 설치, 활성화, 비활성화 또는 제거합니다.	semodule

rhel-system-roles 패키지에서 설치한 `/usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml` 예제 플레이북은 강제 모드에서 대상 정책을 설정하는 방법을 보여줍니다. 또한 플레이북은 `/tmp/test_dir/` 디렉터리의 여러 로컬 정책 수정 및 복원 파일 컨텍스트를 적용합니다.

SELinux 역할 변수에 대한 자세한 참조는 **rhel-system-roles** 패키지를 설치하고 `/usr/share/doc/rhel-system-roles/selinux/` 디렉터리에서 **README.md** 또는 **README.html** 파일을 참조하십시오.

추가 리소스

- [RHEL 시스템 역할 소개.](#)

9.2. SELINUX 시스템 역할을 사용하여 여러 시스템에 SELINUX 설정 적용

단계에 따라 검증된 **SELinux** 설정으로 **Ansible** 플레이북을 준비 및 적용합니다.

사전 요구 사항

- **SELinux** 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 **Ansible Core**(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

•

관리 노드를 나열하는 인벤토리 파일.

절차

1.

플레이북을 준비합니다. 처음부터 시작하거나 **rhel-system-roles** 패키지의 일부로 설치된 예제 플레이북을 수정할 수 있습니다.

```
# cp /usr/share/doc/rhel-system-roles/selinux/example-selinux-playbook.yml my-selinux-playbook.yml
# vi my-selinux-playbook.yml
```

2.

시나리오에 맞게 플레이북의 내용을 변경합니다. 예를 들어 다음 부분에서는 시스템이 **selinux-local-1.pp SELinux** 모듈을 설치하고 활성화합니다.

```
selinux_modules:
- { path: "selinux-local-1.pp", priority: "400" }
```

3.

변경 사항을 저장하고 텍스트 편집기를 종료합니다.

4.

host1, **host2** 및 **host3** 시스템에서 플레이북을 실행합니다.

```
# ansible-playbook -i host1,host2,host3 my-selinux-playbook.yml
```

추가 리소스

- 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/selinux/** 및 **/usr/share/ansible/roles/rhel-system-roles.selinux/** 디렉터리를 참조하십시오.

10장. 로깅 시스템 역할 사용

시스템 관리자는 로깅 시스템 역할을 사용하여 여러 클라이언트 시스템에서 로그를 수집하도록 RHEL 호스트를 로깅 서버로 구성할 수 있습니다.

10.1. 로깅 시스템 역할

로깅 시스템 역할을 사용하면 로컬 및 원격 호스트에 로깅 구성을 배포할 수 있습니다.

하나 이상의 시스템에 로깅 시스템 역할을 적용하려면 *플레이북*에서 로깅 구성을 정의합니다. 플레이북은 하나 이상의 플레이 목록입니다. 플레이북은 사람이 읽을 수 있으며 **YAML** 형식으로 작성됩니다. 플레이북에 대한 자세한 내용은 **Ansible** 설명서의 [플레이북 작업](#)을 참조하십시오.

플레이북에 따라 구성하려는 시스템 집합은 *인벤토리 파일*에 정의되어 있습니다. 인벤토리 생성 및 사용에 대한 자세한 내용은 **Ansible** 문서에서 [인벤토리를 빌드하는 방법](#)을 참조하십시오.

로깅 솔루션은 로그를 읽는 다양한 방법과 여러 로깅 출력을 제공합니다.

예를 들어 로깅 시스템은 다음과 같은 입력을 수신할 수 있습니다.

- 로컬 파일,
- **systemd/journal**.
- 네트워크를 통한 다른 로깅 시스템입니다.

또한 로깅 시스템에는 다음 출력이 있을 수 있습니다.

- **/var/log** 디렉토리의 로컬 파일에 저장된 로그
- **Elasticsearch**로 전송되는 로그

- 다른 로깅 시스템으로 전달된 로그입니다.

로깅 시스템 역할을 사용하면 입력 및 출력을 결합하여 시나리오에 적합할 수 있습니다. 예를 들어 저널의 입력을 로컬 파일에 저장하는 로깅 솔루션을 구성할 수 있지만, 파일에서 읽은 입력은 다른 로깅 시스템으로 전달되고 로컬 로그 파일에 저장됩니다.

10.2. 로깅 시스템 역할 매개변수

로깅 시스템 역할 플레이북에서는 **logging_inputs** 매개변수에 입력을 정의하고, **logging_outputs** 매개변수의 출력 및 **logging_flows** 매개변수의 입력과 출력 간의 관계를 정의합니다. 로깅 시스템 역할은 로깅 시스템을 구성하는 추가 옵션으로 이러한 변수를 처리합니다. 또한 암호화를 활성화할 수도 있습니다.



참고

현재 로깅 시스템 역할에서 사용 가능한 유일한 로깅 시스템은 **Rsyslog**입니다.

- **logging_inputs:** 로깅 솔루션의 입력 목록입니다.
 - 이름: 입력의 고유 이름입니다. **logging_flows:** 입력 목록 및 생성된 구성 파일 이름의 일부입니다.
 - 유형: 입력 요소의 형식입니다. **The type of the input element.** 유형은 **roles/rsyslog/{tasks,vars}/inputs/**의 디렉터리 이름에 해당하는 작업 유형을 지정합니다.
- 기본 사항: **systemd journal** 또는 **unix** 소켓에서 입력을 구성하는 입력입니다.
 - **kernel_message:** **true**로 설정된 경우 **imklog**를 로드합니다. 기본값은 **false**입니다.
 - **use_imuxsock:** **imuxsock** 대신 **imuxsock**을 사용합니다. 기본값은 **false**입니다.
 - **ratelimit_burst:** **ratelimit_interval**으로 내보낼 수 있는 최대 메시지 수입니다.

- 출력: `logging_outputs` 이름 값 목록입니다.

추가 리소스

- `/usr/share/ansible/roles/ rhel-system-roles.logging/README.html`의 `rhel-system-roles` 패키지와 함께 설치된 설명서

10.3. 로컬 로깅 시스템 역할 적용

다음 단계에 따라 별도의 시스템 세트에서 로깅 솔루션을 구성하는 **Ansible** 플레이북을 준비 및 적용합니다. 각 머신은 로컬로 로그를 기록합니다.

사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- `ansible-core` 및 `rhel-system-roles` 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 `ansible`, `ansible-playbook`, `docker` 및 `podman` 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core**(`ansible-core` 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원](#) 범위를 참조하십시오.

•

관리 노드를 나열하는 인벤토리 파일.



참고

시스템 역할이 배포 시 **rsyslog** 패키지를 설치하므로 **rsyslog** 패키지가 설치되어 있지 않아도 됩니다.

절차

1.

필요한 역할을 정의하는 플레이북을 생성합니다.

a.

새 **YAML** 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

```
# vi logging-playbook.yml
```

b.

다음 콘텐츠를 삽입합니다.

```
---
- name: Deploying basics input and implicit files output
  hosts: all
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: system_input
        type: basics
    logging_outputs:
      - name: files_output
        type: files
    logging_flows:
      - name: flow1
        inputs: [system_input]
        outputs: [files_output]
```

2.

특정 인벤토리에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory-file /path/to/file/logging-playbook.yml
```

다음과 같습니다.

- **inventory-file** 은 인벤토리 파일입니다.
- **logging-playbook.yml** 은 사용하는 플레이북입니다.

검증

1. **/etc/journal.conf** 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 시스템이 로그에 메시지를 전송하는지 확인합니다.

- a. 테스트 메시지를 보냅니다.

```
# logger test
```

- b. **/var/log/messages** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/messages
Aug 5 13:48:31 hostname root[6778]: test
```

여기서 '**hostname**'은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 루트).

10.4. 로컬 로깅 시스템 역할에서 로그 필터링

rsyslog 속성 기반 필터를 기반으로 로그를 필터링하는 로깅 솔루션을 배포할 수 있습니다.

사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 **관리형 노드**에 대한 액세스 및 권한.

•

Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 **제어** 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

◦

Red Hat Ansible Core 설치

◦

rhel-system-roles 패키지가 설치되어 있습니다.

◦

관리 노드를 나열하는 인벤토리 파일.



참고

시스템 역할은 배포 시 **rsyslog** 를 설치하기 때문에 **rsyslog** 패키지를 설치할 필요가 없습니다.

절차

1.

다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploying files input and configured files output
  hosts: all
  roles:
    - linux-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input
        type: basics
    logging_outputs:
      - name: files_output0
        type: files
        property: msg
        property_op: contains
        property_value: error
        path: /var/log/errors.log
      - name: files_output1
        type: files
        property: msg
        property_op: "!contains"
        property_value: error
        path: /var/log/others.log
    logging_flows:
```

```
- name: flow0
  inputs: [files_input]
  outputs: [files_output0, files_output1]
```

이 구성을 사용하면 오류 문자열이 포함된 모든 메시지가 **/var/log/errors.log**에 기록되고 기타 모든 메시지는 **/var/log/others.log**에 기록됩니다.

error 속성 값을 필터링하려는 문자열로 교체할 수 있습니다.

환경 설정에 따라 변수를 수정할 수 있습니다.

2.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

검증

1.

/etc/journal.conf 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2.

시스템에서 오류 문자열이 포함된 메시지를 로그에 전송하는지 확인합니다.

a.

테스트 메시지를 보냅니다.

```
# logger error
```

b.

/var/log/errors.log 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/errors.log
Aug  5 13:48:31 hostname root[6778]: error
```

여기서 **hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 루트).

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.logging/README.html](#)의 **rhel-system-roles** 패키지와 함께 설치된 설명서

10.5. 로깅 시스템 역할을 사용하여 원격 로깅 솔루션 적용

다음 단계에 따라 **Red Hat Ansible Core** 플레이북을 준비 및 적용하여 원격 로깅 솔루션을 구성합니다. 이 플레이북에서 하나 이상의 클라이언트는 **systemd-journal** 에서 로그를 가져와 원격 서버로 전달합니다. 서버는 **remote_rsyslog** 및 **remote_files** 에서 원격 입력을 수신하고 원격 호스트 이름으로 이름이 지정된 디렉토리의 로컬 파일에 로그를 출력합니다.

사전 요구 사항

- 로깅 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.



참고

시스템 역할은 배포 시 **rsyslog** 를 설치하기 때문에 **rsyslog** 패키지를 설치할 필요가 없습니다.

절차

1.

필요한 역할을 정의하는 플레이북을 생성합니다.

a.

새 **YAML** 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

```
# vi logging-playbook.yml
```

b.

파일에 다음 내용을 삽입합니다.

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: remote_udp_input
        type: remote
        udp_ports: [ 601 ]
      - name: remote_tcp_input
        type: remote
        tcp_ports: [ 601 ]
    logging_outputs:
      - name: remote_files_output
        type: remote_files
    logging_flows:
      - name: flow_0
        inputs: [remote_udp_input, remote_tcp_input]
        outputs: [remote_files_output]

- name: Deploying basics input and forwards output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: forward_output0
        type: forwards
        severity: info
        target: _host1.example.com_
        udp_port: 601
      - name: forward_output1
        type: forwards
        facility: mail
        target: _host1.example.com_
        tcp_port: 601
    logging_flows:
      - name: flows0
```

```
inputs: [basic_input]
outputs: [forward_output0, forward_output1]
```

```
[basic_input]
[forward_output0, forward_output1]
```

여기서 **host1.example.com** 은 로깅 서버입니다.



참고

필요에 맞게 플레이북의 매개변수를 수정할 수 있습니다.



주의

로깅 솔루션은 서버 또는 클라이언트 시스템의 **SELinux** 정책에 정의된 포트에서만 작동하며 방화벽에서 열립니다. 기본 **SELinux** 정책에는 포트 **601, 514, 6514, 10514** 및 **20514**가 포함됩니다. 다른 포트를 사용하려면 **클라이언트 및 서버 시스템에서 SELinux 정책을 수정** 합니다. 시스템 역할을 통한 방화벽 구성은 아직 지원되지 않습니다.

2.

서버 및 클라이언트를 나열하는 인벤토리 파일을 생성합니다.

a.

새 파일을 생성하고 텍스트 편집기에서 엽니다. 예를 들면 다음과 같습니다.

```
# vi inventory.ini
```

b.

인벤토리 파일에 다음 내용을 삽입합니다.

```
[servers]
server ansible_host=host1.example.com
[clients]
client ansible_host=host2.example.com
```

다음과 같습니다.

- ***host1.example.com*** 은 로깅 서버입니다.
- ***host2.example.com*** 은 로깅 클라이언트입니다.

3. 인벤토리에서 플레이북을 실행합니다.

```
# ansible-playbook -i /path/to/file/inventory.ini /path/to/file/_logging-playbook.yml
```

다음과 같습니다.

- ***inventory.ini*** 는 인벤토리 파일입니다.
- ***logging-playbook.yml*** 은 사용자가 생성한 플레이북입니다.

검증

1. 클라이언트 및 서버 시스템에서 ***/etc/rsyslog.conf*** 파일의 구문을 테스트합니다.

```
# rsyslogd -N 1
rsyslogd: version 8.1911.0-6.el8, config validation run (level 1), master config
/etc/rsyslog.conf
rsyslogd: End of config validation run. Bye.
```

2. 클라이언트 시스템이 서버에 메시지를 전송하는지 확인합니다.

- a. 클라이언트 시스템에서 테스트 메시지를 보냅니다.

```
# logger test
```

- b. 서버 시스템에서 ***/var/log/messages*** 로그를 확인합니다. 예를 들면 다음과 같습니다.

```
# cat /var/log/messages
Aug 5 13:48:31 host2.example.com root[6778]: test
```

여기서 **host2.example.com** 은 클라이언트 시스템의 호스트 이름입니다. 로그에는 **logger** 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 루트).

추가 리소스

- [RHEL 시스템 역할 시작하기](#)
- [/usr/share/ansible/roles/rhel-system-roles.logging/README.html](#)의 **rhel-system-roles**패키지와 함께 설치된 설명서
- [RHEL 시스템 역할 KB 문서](#)

10.6. TLS를 사용하여 로깅 시스템 역할 사용

TLS(Transport Layer Security)는 컴퓨터 네트워크를 통해 안전하게 통신하도록 설계된 암호화 프로토콜입니다.

관리자는 로깅 **RHEL** 시스템 역할을 사용하여 **Red Hat Ansible Automation Platform**을 사용하여 로그의 보안 전송을 구성할 수 있습니다.

10.6.1. TLS로 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 로그인한 **RHEL** 시스템에서 로깅을 구성하고 **Ansible** 플레이북을 실행하여 **TLS**를 사용하여 로그를 원격 로깅 시스템으로 전송할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에 **TLS**를 구성합니다. **TLS** 프로토콜은 네트워크를 통해 로그를 안전하게 전송하기 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- **TLS**를 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.
- 관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.

ansible 및 rhel-system-roles 패키지는 제어 노드에 설치됩니다.

절차

1.

다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploying files input and forwards output with certs
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: files
        input_log_path: /var/log/containers/*.log
    logging_outputs:
      - name: output_name
        type: forwards
        target: your_target_host
        tcp_port: 514
        tls: true
        pki_authmode: x509/name
        permitted_server: 'server.example.com'
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

플레이북은 다음 매개 변수를 사용합니다.

logging_pki_files

이 매개변수를 사용하면 TLS를 구성할 수 있으며 **ca_cert_src**, **cert_src**, **private_key_src** 매개변수를 전달해야 합니다.

ca_cert

CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

cert

인증서 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일

이름은 사용자가 설정합니다.

private_key

개인 키의 경로를 나타냅니다. **Represents the path to private key.** 기본 경로는 `/etc/pki/tls/private/server-key.pem`이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 해당 위치에 복사됩니다.

cert_src

대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert** 가 지정되면 해당 위치에 복사됩니다.

private_key_src

대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 를 지정하면 위치에 복사됩니다.

tls

이 매개변수를 사용하면 네트워크를 통해 안전한 로그를 전송할 수 있습니다. 보안 래퍼를 원하지 않는 경우 **tls: true** 를 설정할 수 있습니다.

2.

플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file playbook.yml
```

10.6.2. TLS를 사용하여 서버 로깅 구성

로깅 시스템 역할을 사용하여 **RHEL** 시스템에서 서버로 로깅을 구성하고 **Ansible** 플레이북을 실행하여 **TLS**가 있는 원격 로깅 시스템에서 로그를 수신할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에 **TLS**를 구성합니다.

사전 요구 사항

- **TLS**를 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.
- 관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.

절차

1. 다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploying remote input and remote_files output with certs
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_pki_files:
      - ca_cert_src: /local/path/to/ca_cert.pem
        cert_src: /local/path/to/cert.pem
        private_key_src: /local/path/to/key.pem
    logging_inputs:
      - name: input_name
        type: remote
        tcp_ports: 514
        tls: true
        permitted_clients: ['clients.example.com']
    logging_outputs:
      - name: output_name
        type: remote_files
        remote_log_path: /var/log/remote/%FROMHOST%/PROGRAMNAME:::secpath-
replace%.log
        async_writing: true
        client_count: 20
        io_buffer_size: 8192
    logging_flows:
      - name: flow_name
        inputs: [input_name]
        outputs: [output_name]
```

플레이북은 다음 매개 변수를 사용합니다.

logging_pki_files

이 매개 변수를 사용하면 **TLS**를 구성할 수 있으며 **ca_cert_src**, **cert_src**,

private_key_src 매개변수를 전달해야 합니다.

ca_cert

CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.

cert

인증서 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.

private_key

개인 키의 경로를 나타냅니다. **Represents the path to private key.** 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.

ca_cert_src

대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다. **ca_cert** 를 지정하면 해당 위치에 복사됩니다.

cert_src

대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert** 가 지정되면 해당 위치에 복사됩니다.

private_key_src

대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 를 지정하면 위치에 복사됩니다.

tls

이 매개변수를 사용하면 네트워크를 통해 안전한 로그를 전송할 수 있습니다. 보안 래퍼를 원하지 않는 경우 **tls: true** 를 설정할 수 있습니다.

2.

플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file playbook.yml
```

10.7. RELP에서 로깅 시스템 역할 사용

REL(Reliable Event Logging Protocol)은 **TCP** 네트워크를 통한 데이터 및 메시지 로깅을 위한 네트워크 위킹 프로토콜입니다. 이벤트 메시지를 안정적으로 전달하고 메시지 손실을 허용하지 않는 환경에서 사용할 수 있습니다.

REL 발신자는 로그 항목을 명령 형태로 전송하고 수신자는 처리되면 이를 승인합니다. 일관성을 보장하기 위해 **REL**은 모든 종류의 메시지 복구를 위해 각 전송된 각 명령에 트랜잭션 번호를 저장합니다.

REL 클라이언트와 **REL** 서버 간의 원격 로깅 시스템을 고려할 수 있습니다. **REL Client**는 원격 로깅 시스템으로 로그를 전송하고 **REL** 서버는 원격 로깅 시스템에서 보낸 모든 로그를 수신합니다.

관리자는 로깅 시스템 역할을 사용하여 로그 항목을 안정적으로 전송하고 수신하도록 로깅 시스템을 구성할 수 있습니다.

10.7.1. RELP로 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 로그인한 **RHEL** 시스템에서 로깅을 구성하고 **Ansible** 플레이북을 실행하여 **REL**을 사용하여 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에 **REL**을 구성합니다. **REL** 구성은 **TLS(Transport Layer Security)**를 사용하여 네트워크를 통해 로그를 안전하게 전송하기 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- **REL**을 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.
- 관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.

절차

1. 다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```

---
- name: Deploying basic input and relp output
  hosts: clients
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: basic_input
        type: basics
    logging_outputs:
      - name: relp_client
        type: relp
        target: _logging.server.com_
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/client-cert.pem_
        private_key: _/etc/pki/tls/private/client-key.pem_
        pki_authmode: name
        permitted_servers:
          - '*.server.example.com'
    logging_flows:
      - name: _example_flow_
        inputs: [basic_input]
        outputs: [relp_client]

```

플레이북은 다음 설정을 사용합니다.

- 대상: 이는 원격 로깅 시스템이 실행 중인 호스트 이름을 지정하는 필수 매개변수입니다.
- 포트: 원격 로깅 시스템이 수신하고 있는 포트 번호입니다.
- **tls:** 네트워크를 통해 안전한 로그 전송을 보장합니다. 보안 래퍼를 원하지 않으면 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**와 함께 작업할 때 **true**로 설정되고 키/인증서 **{ca_cert,private_key}** 및/또는 **{ca_cert_src,cert_src,cert_src}**가 필요합니다.
 - **{ca_cert_src,cert_src,private_key_src}** 트립릿이 설정된 경우 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 이 제어 노드에서 파일을 전송하는 대상으로 사용됩니다. 이 경우 파일 이름은 트리플릿의 원래 이름과 동일합니다.
 - **{ca_cert,cert,private_key}** 트립릿이 설정된 경우 로깅 구성보다 먼저 파일이 기본 경로에 있어야 합니다.

- 트리플릿이 모두 설정되면 파일은 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.
- **ca_cert**: CA 인증서의 경로를 나타냅니다. 기본 경로는 `/etc/pki/tls/certs/ca.pem` 이며 파일 이름은 사용자가 설정합니다.
- **cert**: 인증서 경로를 나타냅니다. 기본 경로는 `/etc/pki/tls/certs/server-cert.pem` 이며 파일 이름은 사용자가 설정합니다.
- **private_key**: 개인 키의 경로를 나타냅니다. **Represents the path to private key.** 기본 경로는 `/etc/pki/tls/private/server-key.pem` 이며 파일 이름은 사용자가 설정합니다.
- **ca_cert_src**: 대상 호스트에 복사되는 로컬 CA 인증서 파일 경로를 나타냅니다. **ca_cert**를 지정하면 해당 위치에 복사됩니다.
- **cert_src**: 대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert**가 지정되면 해당 위치에 복사됩니다.
- **private_key_src**: 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key**를 지정하면 위치에 복사됩니다.
- **pki_authmode**: 인증 모드를 이름 또는 지문으로 허용합니다.
- **permitted_servers**: 로깅 클라이언트에서 TLS를 통해 연결하고 로그를 보내도록 허용할 서버 목록입니다.
- 입력: 로깅 입력 사전 목록입니다.
- 출력: 로깅 출력 사전 목록입니다.

2.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file playbook.yml
```

10.7.2. RELP로 서버 로깅 구성

Logging System Role을 사용하여 **RHEL** 시스템의 로깅을 서버로 구성하고 **Ansible** 플레이북을 실행하여 **RELP**로 원격 로깅 시스템에서 로그를 수신할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에 **RELP**를 구성합니다. **RELP** 구성은 **TLS**를 사용하여 네트워크를 통해 로그를 안전하게 전송하기 위해 메시지 전송을 암호화합니다.

사전 요구 사항

- **RELP**를 구성할 관리 노드에서 플레이북을 실행할 수 있는 권한이 있습니다.
- 관리형 노드는 제어 노드의 인벤토리 파일에 나열됩니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.

절차

1.

다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploying remote input and remote_files output
  hosts: server
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: relp_server
        type: relp
        port: 20514
        tls: true
        ca_cert: _/etc/pki/tls/certs/ca.pem_
        cert: _/etc/pki/tls/certs/server-cert.pem_
        private_key: _/etc/pki/tls/private/server-key.pem_
        pki_authmode: name
        permitted_clients:
```

```

- '_example.client.com_'
logging_outputs:
- name: _remote_files_output_
  type: _remote_files_
logging_flows:
- name: _example_flow_
  inputs: _relp_server_
  outputs: _remote_files_output_

```

플레이북은 다음 설정을 사용합니다.

- 포트: 원격 로깅 시스템이 수신하고 있는 포트 번호입니다.
- - **tls:** 네트워크를 통해 안전한 로그 전송을 보장합니다. 보안 레퍼를 원하지 않으면 **tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로 **tls** 매개변수는 **RELP**와 함께 작업할 때 **true**로 설정되고 키/인증서 **{ca_cert,private_key}** 및/또는 **{ca_cert_src },cert_src, cert_src}**가 필요합니다.
 - **{ca_cert_src,cert_src,private_key_src}** 트립릿이 설정된 경우 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 이 제어 노드에서 파일을 전송하는 대상으로 사용됩니다. 이 경우 파일 이름은 트리플릿의 원래 이름과 동일합니다.
 - **{ca_cert,cert,private_key}** 트립릿이 설정된 경우 로깅 구성보다 먼저 파일이 기본 경로에 있어야 합니다.
 - 트리플릿이 모두 설정되면 파일은 제어 노드에서 관리 노드의 특정 경로로 전송됩니다.
- **ca_cert:** CA 인증서의 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.
- **cert:** 인증서 경로를 나타냅니다. 기본 경로는 **/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.
- **private_key:** 개인 키의 경로를 나타냅니다. Represents the path to private key. 기본 경로는 **/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.
- **ca_cert_src:** 대상 호스트에 복사되는 로컬 **CA** 인증서 파일 경로를 나타냅니다.

ca_cert를 지정하면 해당 위치에 복사됩니다.

- **cert_src**: 대상 호스트에 복사되는 로컬 인증서 파일 경로를 나타냅니다. **cert**가 지정되면 해당 위치에 복사됩니다.
- **private_key_src**: 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key**를 지정하면 위치에 복사됩니다.
- **pki_authmode**: 인증 모드를 이름 또는 지문으로 허용합니다.
- **permitted_clients**: 로깅 서버에서 TLS를 통해 로그를 연결하고 보내는 데 허용할 클라이언트 목록입니다.
- **입력**: 로깅 입력 사전 목록입니다.
- **출력**: 로깅 출력 사전 목록입니다.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file playbook.yml
```

10.8. 추가 리소스

- **RHEL 시스템 역할 시작하기**
- `/usr/share/ansible/roles/rhel-system-roles.logging/README.html`의 **rhel-system-roles** 패키지와 함께 설치된 설명서입니다.

- **RHEL 시스템 역할**
- **ansible-playbook(1) 도움말 페이지**

11장. SSH 시스템 역할을 사용하여 보안 통신 구성

관리자는 **SSHD** 시스템 역할을 사용하여 **SSH** 서버와 **SSH** 시스템 역할을 구성하여 **Ansible Core** 패키지를 사용하여 여러 **RHEL** 시스템에서 동시에 **SSH** 클라이언트를 일관되게 구성할 수 있습니다.

11.1. SSH SERVER 시스템 역할 변수

SSH Server 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 **SSH** 구성 파일의 매개 변수를 정의할 수 있습니다.

이러한 변수를 구성하지 않으면 시스템 역할은 **RHEL** 기본값과 일치하는 **sshd_config** 파일을 생성합니다.

모든 경우에 부울이 **sshd** 구성에서 **yes** 및 **no**로 올바르게 렌더링됩니다. 목록을 사용하여 여러 줄 구성 항목을 정의할 수 있습니다. 예를 들면 다음과 같습니다.

```
sshd_ListenAddress:
- 0.0.0.0
- '::'
```

다음과 같이 렌더링됩니다.

```
ListenAddress 0.0.0.0
ListenAddress ::
```

SSH 서버 시스템 역할에 대한 변수

sshd_enable

False로 설정하면 역할이 완전히 비활성화됩니다. 기본값은 **True**입니다.

sshd_skip_defaults

True로 설정하면 시스템 역할이 기본값이 적용되지 않습니다. 대신 **sshd dict** 또는 **sshd_Key** 변수를 사용하여 전체 구성 기본값 세트를 지정합니다. 기본값은 **False**입니다.

sshd_manage_service

False로 설정하면 서비스가 관리되지 않으므로 부팅 시 활성화되지 않으며 시작 또는 다시 로드되지 않습니다. **Ansible service** 모듈이 현재 **AIX**에 대해 **enabled**되지 않으므로 컨테이너 또는 **AIX** 내부에서 실행되는 경우를 제외하고 기본값은 **True**입니다.

sshd_allow_reload

False로 설정하면 구성이 변경된 후 **sshd**가 다시 로드되지 않습니다. 이는 문제 해결에 도움이 될 수 있습니다. 변경된 구성을 적용하려면 **sshd**를 수동으로 다시 로드합니다. 기본값은 **AIX**를 제외하고 **sshd_manage_service**와 동일한 값입니다. 여기서 **sshd_manage_service** 기본값은 **False**이지만 **sshd_allow_reload**의 기본값은 **True**입니다.

sshd_install_service

True로 설정하면 역할은 **sshd** 서비스에 대한 서비스 파일을 설치합니다. 이렇게 하면 운영 체제에 제공된 파일이 재정의됩니다. 두 번째 인스턴스를 구성하고 **sshd_service** 변수도 변경하지 않는 한 **True**로 설정하지 마십시오. 기본값은 **False**입니다.

역할은 다음 변수에서 가리키는 파일을 템플릿으로 사용합니다.

```
sshd_service_template_service (default: templates/sshd.service.j2)
sshd_service_template_at_service (default: templates/sshd@.service.j2)
sshd_service_template_socket (default: templates/sshd.socket.j2)
```

sshd_service

이 변수는 두 번째 **sshd** 서비스 인스턴스를 구성하는 데 유용한 **sshd** 서비스 이름을 변경합니다.

sshd

구성이 포함된 **dict**입니다. 예를 들면 다음과 같습니다.

```
sshd:
  Compression: yes
  ListenAddress:
    - 0.0.0.0
```

sshd_OptionName

dict 대신 **sshd_** 접두사와 옵션 이름으로 구성된 간단한 변수를 사용하여 옵션을 정의할 수 있습니다. **simple** 변수는 **sshd dict**의 값을 재정의합니다. 예를 들면 다음과 같습니다.

```
sshd_Compression: no
```

sshd_match 및 sshd_match_1 to sshd_match_9

dicts 목록 또는 일치 섹션의 경우 **dict**에 불과합니다. 이러한 변수는 **sshd dict**에 정의된 대로 일치하는 블록을 재정의하지 않습니다. 결과 구성 파일에 모든 소스가 반영됩니다.

SSH 서버 시스템 역할에 대한 보조 변수

이러한 변수를 사용하여 지원되는 각 플랫폼에 해당하는 기본값을 재정의할 수 있습니다.

sshd_packages

이 변수를 사용하여 설치된 패키지의 기본 목록을 재정의할 수 있습니다.

sshd_config_owner, sshd_config_group, and sshd_config_mode

이 역할에서 이러한 변수를 사용하여 생성하는 **openssh** 구성 파일의 소유권 및 권한을 설정할 수 있습니다.

sshd_config_file

이 역할이 **openssh** 서버 구성이 생성된 경로입니다.

sshd_config_namespace

이 변수의 기본값은 **null**입니다. 즉, 역할이 시스템 기본값을 포함하여 구성 파일의 전체 콘텐츠를 정의함을 의미합니다. 또는 이 변수를 사용하여 드롭인 디렉터리를 지원하지 않는 시스템의 단일 플레이북에 있는 다른 역할 또는 여러 위치에서 이 역할을 호출할 수 있습니다. **sshd_skip_defaults** 변수는 무시되며 이 경우 시스템 기본값이 사용되지 않습니다.

이 변수를 설정하면 역할이 지정된 구성을 지정된 네임스페이스 아래에 기존 구성 파일의 구성 스택에 배치합니다. 시나리오에 역할을 여러 번 적용해야 하는 경우 각 애플리케이션에 대해 다른 네임스페이스를 선택해야 합니다.



참고

openssh 구성 파일의 제한 사항은 계속 적용됩니다. 예를 들어 구성 파일에 지정된 첫 번째 옵션만 대부분의 구성 옵션에 적용됩니다.

기술적으로, 역할은 기존 구성 파일의 이전 일치 블록과 관계없이 적용되도록 다른 일치 블록을 포함하지 않는 한 **"Match all"** 블록에 코드 조각을 배치합니다. 이를 통해 다양한 역할 호출에서 충돌하지 않는 옵션을 구성할 수 있습니다.

sshd_binary

openssh의 **sshd** 실행 파일의 경로입니다.

sshd_service

sshd 서비스의 이름입니다. 기본적으로 이 변수에는 대상 플랫폼에서 사용하는 **sshd** 서비스의 이름이 포함됩니다. 또한 역할에서 **sshd_install_service** 변수를 사용하는 경우 사용자 지정 **sshd** 서

비스의 이름을 설정할 수도 있습니다.

sshd_verify_hostkeys

기본값은 **auto** 입니다. **auto**로 설정하면 생성된 구성 파일에 있는 모든 호스트 키가 나열되고 존재하지 않는 경로가 생성됩니다. 또한 권한 및 파일 소유자는 기본값으로 설정됩니다. 이 기능은 배포 단계에서 역할을 사용하여 첫 번째 시도에서 서비스를 시작할 수 있는지 확인하는 데 유용합니다. 이 확인을 비활성화하려면 이 변수를 빈 목록 []으로 설정합니다.

sshd_hostkey_owner, sshd_hostkey_group, sshd_hostkey_mode

이러한 변수를 사용하여 **sshd_verify_hostkeys**에서 호스트 키에 대한 소유권 및 권한을 설정합니다.

sshd_sysconfig

RHEL 기반 시스템에서 이 변수는 **sshd** 서비스에 대한 추가 세부 정보를 구성합니다. **true**로 설정하면 이 역할은 다음 구성에 따라 **/etc/sysconfig/sshd** 구성 파일도 관리합니다. 기본값은 **false**입니다.

sshd_sysconfig_override_crypto_policy

RHEL에서 **true**로 설정하면 이 변수가 시스템 전체 암호화 정책을 재정의합니다. 기본값은 **false**입니다.

sshd_sysconfig_use_strong_rng

RHEL 기반 시스템에서 이 변수는 **sshd**에서 인수로 지정된 바이트 수를 사용하여 **openssl** 임의 번호 생성기를 다시 작성할 수 있습니다. 기본값은 **0**으로, 이 기능을 비활성화합니다. 시스템에 하드웨어 임의 번호 생성기가 없는 경우 이 값을 설정하지 마십시오.

11.2. SSH 서버 시스템 역할을 사용하여 OPENSSH 서버 구성

SSH 서버 시스템 역할을 사용하여 **Ansible** 플레이북을 실행하여 여러 **SSH** 서버를 구성할 수 있습니다.



참고

SSH 및 SSHD 구성을 변경하는 다른 시스템 역할(예: **Identity Management RHEL** 시스템 역할)과 함께 **SSH** 서버 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않도록 하려면 **SSH** 서버 역할에서 네임스페이스(RHEL 8 및 이전 버전) 또는 드롭인 디렉터리(RHEL 9)를 사용하는지 확인하십시오.

사전 요구 사항

- **SSHD** 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core**(**ansible-core** 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위를 참조하십시오](#).

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. **SSH** 서버 시스템 역할에 대한 예제 플레이북을 복사합니다.

```
# cp /usr/share/doc/rhel-system-roles/sshd/example-root-login-playbook.yml path/custom-playbook.yml
```

2. 텍스트 편집기를 사용하여 복사된 플레이북을 엽니다. 예를 들면 다음과 같습니다.

```
# vim path/custom-playbook.yml
---
- hosts: all
```

```

tasks:
- name: Configure sshd to prevent root and password login except from particular subnet
  include_role:
    name: rhel-system-roles.sshd
vars:
  sshd:
    # root login and password login is enabled only from a particular subnet
    PermitRootLogin: no
    PasswordAuthentication: no
  Match:
    - Condition: "Address 192.0.2.0/24"
      PermitRootLogin: yes
      PasswordAuthentication: yes

```

플레이북은 다음을 수행하도록 구성된 **SSH** 서버로 관리 노드를 구성합니다.

- 암호 및 **root** 사용자 로그인이 비활성화되어 있습니다
- 암호 및 **root** 사용자 로그인은 서브넷 **192.0.2.0/24**에서만 활성화됩니다.

환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은 **SSH Server 시스템 역할 변수**를 참조하십시오.

3. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

4. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

...

PLAY RECAP

```
localhost : ok=12 changed=2 unreachable=0 failed=0
skipped=10 rescued=0 ignored=0
```

검증

1.

SSH 서버에 로그인합니다.

```
$ ssh user1@10.1.1.1
```

다음과 같습니다.

- **user1** 은 SSH 서버의 사용자입니다.
- **10.1.1.1** 은 SSH 서버의 IP 주소입니다.

2.

SSH 서버에서 sshd_config 파일의 내용을 확인합니다.

```
$ vim /etc/ssh/sshd_config

# Ansible managed
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE LC_MONETARY
LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL LANGUAGE
AcceptEnv XMODIFIERS
AuthorizedKeysFile .ssh/authorized_keys
ChallengeResponseAuthentication no
GSSAPIAuthentication yes
GSSAPICleanupCredentials no
PasswordAuthentication no
PermitRootLogin no
PrintMotd no
Subsystem sftp /usr/libexec/openssh/sftp-server
SyslogFacility AUTHPRIV
UsePAM yes
X11Forwarding yes
Match Address 192.0.2.0/24
    PasswordAuthentication yes
    PermitRootLogin yes
```

3.

192.0.2.0/24 서브넷에서 root로 서버에 연결할 수 있는지 확인합니다.

a.

IP 주소를 확인합니다.

```
$ hostname -I
192.0.2.1
```

IP 주소가 **192.0.2.1 - 192.0.2.254** 범위 내에 있는 경우 서버에 연결할 수 있습니다.

b.

root로 서버에 연결합니다 :

```
$ ssh root@10.1.1.1
```

추가 리소스

- [/usr/share/doc/rhel-system-roles/sshd/README.md](#) 파일
- [ansible-playbook\(1\)](#) 도움말 페이지

11.3. SSH 클라이언트 시스템 역할 변수

SSH 클라이언트 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 클라이언트 **SSH** 구성 파일의 매개 변수를 정의할 수 있습니다.

이러한 변수를 구성하지 않으면 시스템 역할은 **RHEL** 기본값과 일치하는 글로벌 **ssh_config** 파일을 생성합니다.

모든 경우에 부울이 **ssh** 구성에서 **yes** 또는 **no** 로 올바르게 렌더링됩니다. 목록을 사용하여 여러 줄 구성 항목을 정의할 수 있습니다. 예를 들면 다음과 같습니다.

```
LocalForward:
- 22 localhost:2222
- 403 localhost:4003
```

다음과 같이 렌더링됩니다.

```
LocalForward 22 localhost:2222
LocalForward 403 localhost:4003
```



참고

구성 옵션은 대소문자를 구분합니다.

SSH 클라이언트 시스템 역할에 대한 변수

ssh_user

시스템 역할이 사용자별 구성을 수정하는 기존 사용자 이름을 정의할 수 있습니다. 사용자별 구성은 지정된 사용자의 `~/.ssh/config`에 저장됩니다. 기본값은 모든 사용자에 대한 글로벌 구성을 수정하는 `null`입니다.

ssh_skip_defaults

기본값은 `auto`입니다. `auto`로 설정하면 시스템 역할은 시스템 전체 구성 파일 `/etc/ssh/ssh_config`를 쓰고 여기에 정의된 **RHEL** 기본값을 유지합니다. `ssh_drop_in_name` 변수를 정의하여 드롭인 구성 파일을 생성하면 `ssh_skip_defaults` 변수가 자동으로 비활성화됩니다.

ssh_drop_in_name

시스템 전체 드롭인 디렉터리에 배치되는 드롭인 구성 파일의 이름을 정의합니다. 이름은 `/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf` 템플릿에서 수정할 구성 파일을 참조하는 데 사용됩니다. 시스템이 드롭인 디렉터리를 지원하지 않는 경우 기본값은 `null`입니다. 시스템이 드롭인 디렉터리를 지원하는 경우 기본값은 `00-ansible`입니다.



주의

시스템이 드롭인 디렉터리를 지원하지 않는 경우 이 옵션을 설정하면 플레이가 실패합니다.

제안된 형식은 **NN-name**입니다. 여기서 **NN**은 구성 파일의 순서를 지정하는 데 사용되는 두 자리 숫자이고, **name**은 파일의 콘텐츠 또는 소유자에 대한 설명이 포함된 이름입니다.

ssh

구성 옵션과 해당 값이 포함된 **dict**입니다.

ssh_OptionName

dict 대신 **ssh_** 접두사 및 옵션 이름으로 구성된 간단한 변수를 사용하여 옵션을 정의할 수 있습니다.

다. 단순 변수는 **ssh dict**의 값을 재정의합니다.

ssh_additional_packages

이 역할은 가장 일반적인 사용 사례에 필요한 **openssh** 및 **openssh-clients** 패키지를 자동으로 설치합니다. 추가 패키지를 설치해야 하는 경우(예: 호스트 기반 인증에 **openssh-keysign**) 이 변수에 지정할 수 있습니다.

ssh_config_file

역할이 생성된 구성 파일을 저장하는 경로입니다. 기본값:

- 시스템에 드롭인 디렉터리가 있는 경우 기본값은 **/etc/ssh/ssh_config.d/{ssh_drop_in_name}.conf** 템플릿으로 정의됩니다.
- 시스템에 드롭인 디렉터리가 없으면 기본값은 **/etc/ssh/ssh_config** 입니다.
- **ssh_user** 변수가 정의된 경우 기본값은 **~/.ssh/config** 입니다.

ssh_config_owner, ssh_config_group, ssh_config_mode

생성된 구성 파일의 소유자, 그룹 및 모드입니다. 기본적으로 파일 소유자는 **root:root**이며 모드는 **0644** 입니다. **ssh_user** 가 정의되면 모드는 **0600** 이고 소유자와 그룹은 **ssh_user** 변수에 지정된 사용자 이름에서 파생됩니다.

11.4. SSH 클라이언트 시스템 역할을 사용하여 OPENSSH 클라이언트 구성

SSH 클라이언트 시스템 역할을 사용하여 **Ansible** 플레이북을 실행하여 여러 SSH 클라이언트를 구성할 수 있습니다.



참고

SSH 및 SSHD 구성을 변경하는 다른 시스템 역할(예: **Identity Management RHEL** 시스템 역할)과 함께 SSH 클라이언트 시스템 역할을 사용할 수 있습니다. 구성을 덮어쓰지 않도록 하려면 SSH 클라이언트 역할에서 드롭인 디렉터리(RHEL 8의 기본값)를 사용하는지 확인합니다.

사전 요구 사항

- SSH 클라이언트 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 *관리형 노드*에

대한 액세스 및 권한.

- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 **제어** 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core**(**ansible-core** 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: all
  tasks:
    - name: "Configure ssh clients"
      include_role:
        name: rhel-system-roles.ssh
  vars:
    ssh_user: root
  ssh:
    Compression: true
    GSSAPIAuthentication: no
    ControlMaster: auto
```

```
ControlPath: ~/.ssh/.cm%C
Host:
  - Condition: example
    Hostname: example.com
    User: user1
ssh_FowardX11: no
```

이 플레이북은 다음 구성을 사용하여 관리 노드에서 **root** 사용자의 **SSH** 클라이언트 기본 설정을 구성합니다.

- 압축이 활성화됩니다.
- **ControlMaster** 멀티플렉싱이 **auto** 로 설정되어 있습니다.
- **example.com** 호스트에 연결하는 예제 별칭은 **user1** 입니다.
- 예제 호스트 별칭은 **example.com** 호스트에 대한 연결을 나타내는 **user1** 사용자 이름으로 생성됩니다.
- **X11** 전달이 비활성화되어 있습니다.

선택적으로 환경 설정에 따라 이러한 변수를 수정할 수 있습니다. 자세한 내용은 **SSH 시스템 역할 변수**를 참조하십시오.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check path/custom-playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file path/custom-playbook.yml
```

검증

- 텍스트 편집기에서 **SSH** 구성 파일을 열어 관리 노드에 올바른 구성이 있는지 확인합니다. 예를 들면 다음과 같습니다.

```
# vi ~root/.ssh/config
```

위에 표시된 예제 플레이북의 애플리케이션을 적용한 후에는 구성 파일에 다음 내용이 포함되어야 합니다.

```
# Ansible managed
Compression yes
ControlMaster auto
ControlPath ~/.ssh/.cm%C
ForwardX11 no
GSSAPIAuthentication no
Host example
  Hostname example.com
  User user1
```

11.5. 비독점 구성에 대해 SSH 서버 시스템 역할 사용

일반적으로 **SSH** 서버 시스템 역할을 적용하면 전체 구성을 덮어씁니다. 이전에 구성을 조정한 경우 (예: 다른 시스템 역할 또는 플레이북이 있는 경우) 문제가 될 수 있습니다. 다른 옵션을 그대로 유지하면서 선택된 구성 옵션에 대해서만 **SSH Server** 시스템 역할을 적용하려면 비독점 구성을 사용할 수 있습니다.

RHEL 8 이전 버전에서는 구성 스니펫을 사용하여 비독점 구성을 적용할 수 있습니다. 자세한 내용은 [RHEL 8 문서의 비독점 구성에 대한 SSH 서버 시스템 역할](#) 사용을 참조하십시오.

RHEL 9에서는 드롭인 디렉터리의 파일을 사용하여 비독점 구성을 적용할 수 있습니다. 기본 구성 파일은 드롭인 디렉터리에 이미 `/etc/ssh/sshd_config.d/00-ansible_system_role.conf` 로 배치되어 있습니다.

사전 요구 사항

- **SSH** 서버 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 *관리형 노드*에 대한 액세스 및 권한.
 - **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.
- 제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 패키지가 설치되어 있습니다.
- 관리 노드를 나열하는 인벤토리 파일.
- 다른 **RHEL** 시스템 역할에 대한 플레이북입니다. 자세한 내용은 **역할 적용**을 참조하십시오.

절차

1.

sshd_config_file 변수의 구성 스니펫을 플레이북에 추가합니다.

```
---
- hosts: all
  tasks:
    - name: <Configure sshd to accept some useful environment variables>
      include_role:
        name: rhel-system-roles.sshd
      vars:
        sshd_config_file: /etc/ssh/sshd_config.d/<42-my-application>.conf
        sshd:
          # Environment variables to accept
          AcceptEnv:
            LANG
            LS_COLORS
            EDITOR
```

sshd_config_file 변수에서 **SSH** 서버 시스템 역할이 구성 옵션을 작성하는 **.conf** 파일을 정의합니다.

2자리 접두사를 사용합니다(예: **42-**)를 사용하여 구성 파일이 적용되는 순서를 지정합니다.

인벤토리에 플레이북을 적용하면 역할은 **sshd_config_file** 변수에서 정의한 파일에 다음 구성 옵션을 추가합니다.

```
# Ansible managed
#
AcceptEnv LANG LS_COLORS EDITOR
```

검증

- 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml -i inventory_file
```

추가 리소스

- [/usr/share/doc/rhel-system-roles/sshd/README.md](#) 파일
- [ansible-playbook\(1\)](#) 도움말 페이지

12장. VPN RHEL 시스템 역할을 사용하여 IPSEC을 사용하여 VPN 연결 구성

VPN 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL** 시스템에서 VPN 연결을 구성할 수 있습니다. 이를 사용하여 **host-to-host**, **network-to-network**, **VPN Remote Access Server** 및 메시 구성을 설정할 수 있습니다.

호스트 간 연결의 경우 역할은 필요에 따라 기본 매개 변수를 사용하여 **vpn_connections** 목록에 있는 각 호스트 쌍 간에 **VPN** 터널을 설정합니다. 또는 나열된 모든 호스트 간에 **opportunistic** 메시 구성을 생성하도록 구성할 수 있습니다. 이 역할은 호스트에 있는 호스트의 이름이 **Ansible** 인벤토리에 사용된 호스트의 이름과 동일하고 해당 이름을 사용하여 터널을 구성할 수 있다고 가정합니다.



참고

VPN RHEL 시스템 역할은 현재 **VPN** 공급자로서 **IPsec** 구현인 **Libreswan**만을 지원합니다.

12.1. VPN 시스템 역할을 사용하여 IPSEC으로 호스트 간 VPN 생성

VPN 시스템 역할을 사용하면 인벤토리 파일에 나열된 모든 관리형 노드를 구성하는 제어 노드에서 **Ansible** 플레이북을 실행하여 호스트 간 연결을 구성할 수 있습니다.

사전 요구 사항

- **VPN** 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형 노드*에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어 노드* 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

관리 노드를 나열하는 인벤토리 파일.

절차

1.

다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- name: Host to host VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - hosts:
          managed_node1:
          managed_node2:
```

이 플레이북은 시스템 역할로 자동 생성되는 키와 사전 공유 키 인증을 사용하여 **managed_node1-to-managed_node2** 연결을 구성합니다.

2.

선택 사항: 다음 섹션을 호스트의 **vpn_connections** 목록에 추가하여 인벤토리 파일에 나열되지 않은 외부 호스트로의 연결을 구성합니다.

```
vpn_connections:
  - hosts:
      managed_node1:
      managed_node2:
      external_node:
        hostname: 192.0.2.2
```

이렇게 하면 *managed_node1-to-external_node* 및 *managed_node2-to-external_node*의 두 개의 추가 연결이 구성됩니다.



참고

연결은 관리형 노드에서만 구성되며 외부 노드에는 구성되지 않습니다.

1.

선택 사항: **vpn_connections** 내의 추가 섹션을 사용하여 관리형 노드에 대해 여러 VPN 연결을 지정할 수 있습니다(예: 컨트롤 플레인 및 데이터 플레인).

```
- name: Multiple VPN
  hosts: managed_node1, managed_node2
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - name: control_plane_vpn
        hosts:
          managed_node1:
            hostname: 192.0.2.0 # IP for the control plane
          managed_node2:
            hostname: 192.0.2.1
      - name: data_plane_vpn
        hosts:
          managed_node1:
            hostname: 10.0.0.1 # IP for the data plane
          managed_node2:
            hostname: 10.0.0.2
```

2.

선택 사항: 환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은 `/usr/share/doc/rhel-system-roles/vpn/README.md` 파일을 참조하십시오.

3.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check /path/to/file/playbook.yml -i /path/to/file/inventory_file
```

4.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i /path/to/file/inventory_file /path/to/file/playbook.yml
```

검증

1.

관리형 노드에서 연결이 성공적으로 로드되었는지 확인합니다.

```
# ipsec status | grep connection.name
```

connection.name을 이 노드의 연결 이름으로 교체합니다(예: **managed_node1-to-managed_node2**).

참고

기본적으로 역할은 각 시스템의 관점에서 생성하는 각 연결에 대해 설명이 포함된 이름을 생성합니다. 예를 들어 **managed_node1** 과 **managed_node2** 간의 연결을 생성할 때 **managed_node1** 에서 이 연결의 설명적인 이름은 **managed_node1-to-managed_node2** 이지만, **managed_node2** 에서 연결 이름은 **managed_node2-to-managed_node1** 입니다.

1.

관리형 노드에서 연결이 성공적으로 시작되었는지 확인합니다.

```
# ipsec trafficstatus | grep connection.name
```

2.

선택 사항: 연결이 성공적으로 로드되지 않은 경우 다음 명령을 입력하여 연결을 수동으로 추가합니다. 이렇게 하면 연결 설정 실패 이유를 나타내는 더 구체적인 정보를 제공합니다.

```
# ipsec auto --add connection.name
```

참고

연결을 로드하고 시작하는 동안 발생한 모든 오류는 로그에 보고되며, **/var/log/pluto.log**에서 확인할 수 있습니다. 이러한 로그는 구문 분석하기 어렵기 때문에 대신 표준 출력에서 로그 메시지를 가져오기 위해 수동으로 연결을 추가합니다.

12.2. VPN 시스템 역할을 사용하여 IPSEC을 통한 OPPORTUNISTIC 메시 VPN 연결 생성

VPN 시스템 역할을 사용하여 제어 노드에서 **Ansible** 플레이북을 실행하여 인증에 인증서를 사용하는 **opportunistic** 메시 VPN 연결을 구성할 수 있습니다. 이 VPN은 인벤토리 파일에 나열된 모든 관리형 노드를 구성합니다.

인증서를 사용한 인증은 플레이북에 **auth_method: cert** 매개변수를 정의하여 구성됩니다. VPN 시스

템 역할은 `/etc/ipsec.d` 디렉터리에 정의된 **IPsec** 네트워크 보안 서비스(**NSS**) 암호화 라이브러리가 필요하다고 가정합니다. 기본적으로 노드 이름은 인증서 닉네임으로 사용됩니다. 이 예에서는 **managed_node1** 입니다. 인벤토리에서 **cert_name** 특성을 사용하여 다른 인증서 이름을 정의할 수 있습니다.

다음 예제 절차에서 **Ansible** 플레이북을 실행하는 시스템인 제어 노드는 관리형 노드(**192.0.2.0/24**)와 동일한 클래스의 도메인 간 라우팅(**CIDR**) 번호를 공유하며, **IP** 주소 **192.0.2.7**이 있습니다. 따라서 제어 노드는 **CIDR 192.0.2.0/24**에 대해 자동으로 생성되는 개인 정책에 속합니다.

플레이 중에 **SSH** 연결 손실을 방지하기 위해 제어 노드에 대한 명확한 정책이 정책 목록에 포함됩니다. **CIDR**이 기본값과 같은 정책 목록에도 항목이 있습니다. 이 플레이북은 기본 정책의 규칙을 재정의하여 **private-or-clear** 대신 비공개로 만들기 때문입니다.

사전 요구 사항

- **VPN** 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형 노드*에 대한 액세스 및 권한.
 - 모든 관리형 노드에서 `/etc/ipsec.d` 디렉터리의 **NSS** 데이터베이스에는 피어 인증에 필요한 모든 인증서가 포함되어 있습니다. 기본적으로 노드 이름은 인증서 닉네임으로 사용됩니다.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어 노드* 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1.

다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- name: Mesh VPN
  hosts: managed_node1, managed_node2, managed_node3
  roles:
    - rhel-system-roles.vpn
  vars:
    vpn_connections:
      - opportunistic: true
        auth_method: cert
        policies:
          - policy: private
            cidr: default
          - policy: private-or-clear
            cidr: 198.51.100.0/24
          - policy: private
            cidr: 192.0.2.0/24
          - policy: clear
            cidr: 192.0.2.7/32
```

2.

선택 사항: 환경 설정에 따라 변수를 수정할 수 있습니다. 자세한 내용은 [/usr/share/doc/rhel-system-roles/vpn/README.md](#) 파일을 참조하십시오.

3.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

4.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

12.3. 추가 리소스

- VPN 시스템 역할에 사용되는 매개변수 및 역할에 대한 추가 정보에 대한 자세한 내용은 `/usr/share/doc/rhel-system-roles/vpn/README.md` 파일을 참조하십시오.
- `ansible-playbook` 명령에 대한 자세한 내용은 `ansible-playbook(1)` 도움말 페이지를 참조하십시오.

13장. 시스템 전체에서 사용자 지정 암호화 정책 설정

관리자는 **Cryptographic Policies RHEL** 시스템 역할을 사용하여 **Ansible Core** 패키지를 사용하여 다양한 시스템에서 사용자 지정 암호화 정책을 빠르고 일관되게 구성할 수 있습니다.

13.1. 암호화 정책 시스템 역할 변수 및 사실

암호화 정책 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 암호화 정책 구성 파일의 매개 변수를 정의할 수 있습니다.

변수를 구성하지 않으면 시스템 역할은 시스템을 구성하지 않고 팩트만 보고합니다.

암호화 정책 시스템 역할에 대해 선택한 변수

crypto_policies_policy

시스템 역할이 관리되는 노드에 적용되는 암호화 정책을 결정합니다. 다양한 암호화 정책에 대한 자세한 내용은 [시스템 전체 암호화 정책](#)을 참조하십시오.

crypto_policies_reload

yes로 설정하면 영향을 받는 서비스(현재 **ipsec**, **bind**, **sshd** 서비스)가 **crypto** 정책을 적용한 후 다시 로드합니다. 기본값은 **yes**입니다.

crypto_policies_reboot_ok

yes로 설정하고 시스템 역할이 암호화 정책을 변경한 후 재부팅이 필요한 경우 **crypto_policies_reboot_required**를 **yes**로 설정합니다. 기본값은 **no**입니다.

암호화 정책 시스템 역할에 의해 설정된 사실

crypto_policies_active

현재 선택한 정책을 나열합니다.

crypto_policies_available_policies

시스템에서 사용 가능한 모든 정책을 나열합니다.

crypto_policies_available_subpolicies

시스템에서 사용 가능한 모든 하위 정책을 나열합니다.

추가 리소스

- [사용자 지정 시스템 전체 암호화 정책 생성 및 설정](#)

13.2. 암호화 정책 시스템 역할을 사용하여 사용자 정의 암호화 정책 설정

암호화 정책 시스템 역할을 사용하여 단일 제어 노드에서 일관되게 많은 수의 관리형 노드를 구성할 수 있습니다.

사전 요구 사항

- **Crypto Policies System Role**으로 구성하려는 시스템인 하나 이상의 *관리형 노드*에 대한 액세스 및 사용 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: all
  tasks:
    - name: Configure crypto policies
      include_role:
        name: rhel-system-roles.crypto_policies
  vars:
    - crypto_policies_policy: FUTURE
    - crypto_policies_reboot_ok: true
```

예를 들어 **FUTURE** 값을 기본 암호화 정책으로 교체할 수 있습니다. **DEFAULT**, **LEGACY** 및 **FIPS:OSPP**.

crypto_policies_reboot_ok: true 변수를 사용하면 시스템 역할에서 암호화 정책을 변경한 후 시스템이 재부팅됩니다.

자세한 내용은 [Policies Policies System Role variables and facts](#) 를 참조하십시오.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file playbook.yml
```

검증

1. 제어 노드에서 이름이 (예: **verify_playbook.yml**)인 다른 플레이북을 생성합니다.

```
- hosts: all
  tasks:
    - name: Verify active crypto policy
      include_role:
        name: rhel-system-roles.crypto_policies
```

```
- debug:
  var: crypto_policies_active
```

이 플레이북은 시스템의 구성을 변경하지 않으며, 관리형 노드에서 활성 정책만 보고합니다.

2.

동일한 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file verify_playbook.yml

TASK [debug] *****
ok: [host] => {
  "crypto_policies_active": "FUTURE"
}
```

"crypto_policies_active": 변수는 관리 노드에서 정책을 활성 상태로 표시합니다.

13.3. 추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.crypto_policies/README.md](#) 파일
- [ansible-playbook\(1\)](#) 도움말 페이지
- [RHEL 시스템 역할 설치](#).
- [시스템 역할 적용](#)

14장. CLEVIS 및 TANG 시스템 역할 사용

14.1. CLEVIS 및 TANG 시스템 역할 소개

RHEL 시스템 역할은 여러 **RHEL** 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 **Ansible** 역할 및 모듈의 컬렉션입니다.

Clevis와 **Tang**을 사용하여 **PBD(Policy-Based Decryption)** 솔루션의 자동 배포에 **Ansible** 역할을 사용할 수 있습니다. **rhel-system-roles** 패키지에는 이러한 시스템 역할, 관련 예제 및 참조 문서가 포함되어 있습니다.

네트워크 **Bound** 디스크 암호화 클라이언트 시스템 역할을 사용하면 자동화된 방식으로 여러 **Clevis** 클라이언트를 배포할 수 있습니다. **Network Bound Disk Encryption Client** 역할은 **Tang** 바인딩만 지원하며 현재 **TPM2** 바인딩에는 사용할 수 없습니다.

네트워크 **Bound Disk Encryption Client** 역할에는 **LUKS**를 사용하여 이미 암호화된 볼륨이 필요합니다. 이 역할은 **LUKS** 암호화된 볼륨을 하나 이상의 **NAT(Network-Bound)** 서버 - **Tang** 서버에 바인딩하는 것을 지원합니다. 암호를 사용하여 기존 볼륨 암호화를 유지하거나 제거할 수 있습니다. 암호를 제거한 후 **RuntimeClass**를 사용하여 볼륨 잠금을 해제할 수 있습니다. 이 기능은 시스템을 프로비저닝한 후 제거해야 하는 임시 키 또는 암호를 사용하여 볼륨을 처음 암호화할 때 유용합니다.

암호와 키 파일을 모두 제공하는 경우 역할은 사용자가 먼저 제공한 파일을 사용합니다. 이러한 유효한 항목을 찾지 못하면 기존 바인딩에서 암호를 검색하려고 합니다.

PBD는 바인딩을 슬롯에 대한 장치 매핑으로 정의합니다. 즉, 동일한 장치에 대한 여러 바인딩이 있을 수 있습니다. 기본 슬롯은 슬롯 **1**입니다.

Network Bound Disk Encryption Client 역할은 **state** 변수도 제공합니다. 새 바인딩을 생성하거나 기존 바인딩을 업데이트하려면 현재 값을 사용합니다. **clevis luks bind** 명령과 반대로 **state: present** 를 사용하여 장치 슬롯의 기존 바인딩을 덮어쓸 수도 있습니다. **absent** 값은 지정된 바인딩을 제거합니다.

네트워크 **Bound** 디스크 암호화 서버 시스템 역할을 사용하여 **Tang** 서버를 자동화된 디스크 암호화 솔루션의 일부로 배포하고 관리할 수 있습니다. 이 역할은 다음 기능을 지원합니다.

- **Tang** 키 교체

- **Tang 키 배포 및 백업**

추가 리소스

- **Network-Bound Disk Encryption(NBDE)** 역할 변수에 대한 자세한 참조를 보려면 **rhel-system-roles** 패키지를 설치한 후 `/usr/share/doc/rhel-system-roles/nbde_client/` 및 `/usr/share/doc/rhel-system-roles/nbde_server/` 디렉터리에 **README.md** 및 **README.html** 파일을 참조하십시오.
- 예를 들어 **system-roles** 플레이북에서 **rhel-system-roles** 패키지를 설치하고 `/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/` 디렉터리를 참조하십시오.
- **RHEL 시스템 역할**에 대한 자세한 내용은 **RHEL 시스템 역할 소개**를 참조하십시오.

14.2. 여러 TANG 서버를 설정하는 데 CLEVIS 서버 시스템 역할 사용

단계에 따라 **Tang** 서버 설정이 포함된 **Ansible** 플레이북을 준비하고 적용합니다.

사전 요구 사항

- 하나 이상의 **관리형** 노드인 **에** 대한 액세스 및 권한(for example, **server system role**)을 사용하여 구성하려는 시스템입니다.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 **제어** 노드에 대한 액세스 및 권한.

제어 노드에서 다음이 있어야 합니다.
 - **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 Ansible 기반 자동화를 위해 Ansible Engine 2.9가 포함된 별도의 Ansible 리포지토리에 대한 액세스를 제공했습니다. Ansible Engine에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. Ansible Engine을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 9.0에서는 Ansible 명령줄 유틸리티, 명령 및 소규모의 기본 제공 Ansible 플러그인 세트가 포함된 Ansible Core(**ansible-core** 패키지로 제공)를 도입했습니다. RHEL은 AppStream 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

•

관리 노드를 나열하는 인벤토리 파일.

절차

1.

Tang 서버에 대한 설정이 포함된 플레이북을 준비합니다. 처음부터 시작하거나 `/usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/` 디렉터리에서 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_server/examples/simple_deploy.yml
./my-tang-playbook.yml
```

2.

선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi my-tang-playbook.yml
```

3.

필수 매개 변수를 추가합니다. 다음 예제 **Playbook**은 **Tang** 서버와 키 교체를 배포합니다.

```
---
- hosts: all

vars:
  nbde_server_rotate_keys: yes

roles:
  - rhel-system-roles.nbde_server
```

4.

완료된 플레이북을 적용합니다.

```
# ansible-playbook -i inventory-file my-tang-playbook.yml
```

여기서: * **inventory-file** 은 인벤토리 파일입니다. * **logging-playbook.yml** 은 사용하는 플레이북입니다.

중요

Clevis가 설치된 시스템에서 **grubby** 도구를 사용하여 초기 부팅 중에 **Tang** 핀의 네트워킹을 사용할 수 있습니다.

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

추가 리소스

•

자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/nbde_server/**, **usr/share/ansible/roles/rhel-system-roles.nbde_server/** 디렉터리를 참조하십시오.

14.3. 여러 CLEVIS 클라이언트를 설정하는 데 CLEVIS 클라이언트 시스템 역할 사용

단계에 따라 **Clevis** 클라이언트 설정이 포함된 **Ansible** 플레이북을 준비하고 적용합니다.

참고

Clevis 클라이언트 시스템 역할은 **Tang** 바인딩만 지원합니다. 즉, **TPM2** 바인딩에 사용할 수 없습니다.

사전 요구 사항

•

하나 이상의 관리형 노드에 대한 액세스 및 권한에 액세스할 수 있습니다. 이 노드인 .이(시스템)는 **Clevis** 클라이언트 시스템 역할을 사용하여 구성하려는 시스템입니다.

•

Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 제어 노드 액세스 및 사용 권한.

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- **rhel-system-roles** 패키지는 플레이북을 실행하려는 시스템에 설치됩니다.

절차

1.

Clevis 클라이언트에 대한 설정이 포함된 플레이북을 준비합니다. 처음부터 시작하거나 `/usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/` 디렉터리에서 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/ansible/roles/rhel-system-roles.nbde_client/examples/high_availability.yml
./my-clevis-playbook.yml
```

2.

선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

```
# vi my-clevis-playbook.yml
```

3.

필수 매개 변수를 추가합니다. 다음 예제 **Playbook**은 두 개 이상의 **Tang** 서버를 사용할 수 있는 경우 두 개 이상의 **LUKS** 암호화 볼륨을 자동으로 잠금 해제하도록 **Clevis** 클라이언트를 구성합니다.

```
---
- hosts: all

  vars:
    nbde_client_bindings:
      - device: /dev/rhel/root
        encryption_key_src: /etc/luks/keyfile
    servers:
      - http://server1.example.com
      - http://server2.example.com
    - device: /dev/rhel/swap
      encryption_key_src: /etc/luks/keyfile
    servers:
      - http://server1.example.com
      - http://server2.example.com

  roles:
    - rhel-system-roles.nbde_client
```

4.

완료된 플레이북을 적용합니다.

```
# ansible-playbook -i host1,host2,host3 my-clevis-playbook.yml
```



중요

Clevis가 설치된 시스템에서 **grubby** 도구를 사용하여 초기 부팅 중에 **Tang** 편의 네트워킹을 사용할 수 있습니다.

```
# grubby --update-kernel=ALL --args="rd.neednet=1"
```

추가 리소스

- **EgressIP Client System Role**에 대한 매개변수 및 추가 정보에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles/nbde_client/** 및 **/usr/share/ansible/rhel-system-roles.nbde_client/** 디렉터리를 참조하십시오.

15장. RHEL 시스템 역할을 사용하여 인증서 요청

인증서 문제 및 업데이트 시스템 역할을 사용하여 인증서를 발행 및 관리할 수 있습니다.

이 장에서는 다음 항목에 대해 설명합니다.

- [인증서 시스템 역할](#)
- [인증서 시스템 역할을 사용하여 새 자체 서명 인증서 요청](#)
- [인증서 시스템 역할을 사용하여 IdM CA에서 새 인증서 요청](#)

15.1. 인증서 문제 및 업데이트 시스템 역할

인증서 문제 및 업데이트 시스템 역할을 사용하면 **Ansible Core**를 사용하여 **TLS** 및 **SSL** 인증서를 발행 및 갱신할 수 있습니다.

이 역할은 **certmonger**를 인증서 공급자로 사용하며 현재 자체 서명된 인증서 발행 및 갱신 및 **IdM** 통합 인증 기관(CA)을 지원합니다.

인증서 문제 및 갱신 시스템 역할과 함께 **Ansible** 플레이북에서 다음 변수를 사용할 수 있습니다.

certificate_wait

작업에서 인증서가 발급될 때까지 기다려야 하는지를 지정합니다.

certificate_requests

발행할 각 인증서와 해당 매개변수를 나타냅니다.

추가 리소스

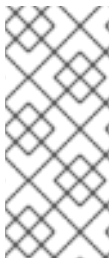
- [/usr/share/ansible/roles/rhel-system-roles.certificate/README.md](#) 파일을 참조하십시오.

- **RHEL 시스템 역할 시작하기**를 참조하십시오.

15.2. 인증서 문제 및 갱신 시스템 역할을 사용하여 새 자체 서명 인증서 요청

인증서 문제 및 업데이트 시스템 역할을 사용하면 **Ansible Core**를 사용하여 자체 서명된 인증서를 발행할 수 있습니다.

이 프로세스에서는 **certmonger** 공급자를 사용하고 **getcert** 명령을 통해 인증서를 요청합니다.



참고

기본적으로 **certmonger**는 만료되기 전에 인증서를 자동으로 업데이트하려고 합니다. **Ansible** 플레이북의 **auto_renew** 매개변수를 **no**로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 **RHEL 시스템 역할 시작하기**를 참조하십시오.

절차

1. **선택 사항:** 인벤토리 파일(예: **inventory.file**)을 생성합니다.

```
$ touch inventory.file
```

2. 인벤토리 파일을 열고 인증서를 요청할 호스트를 정의합니다. 예를 들면 다음과 같습니다.

```
[webserver]
server.idm.example.com
```

3.

플레이북 파일을 만듭니다(예: **request-certificate.yml**):

- 웹 서버와 같이 인증서를 요청할 호스트를 포함하도록 호스트를 설정합니다.
- 다음을 포함하도록 **certificate_requests** 변수를 설정합니다.
 - **name** 매개변수를 **mycert** 와 같은 인증서의 원하는 이름으로 설정합니다.
 - **dns** 매개 변수를 인증서에 포함할 도메인(예: ***.example.com**)으로 설정합니다.
 - **ca** 매개 변수를 자체 서명 으로 설정합니다.
- 역할에 **rhel-system-roles.certificate** 역할을 설정합니다.

이 예제는 플레이북 파일입니다.

```
---
- hosts: webserver

  vars:
    certificate_requests:
      - name: mycert
        dns: "*.example.com"
        ca: self-sign

  roles:
    - rhel-system-roles.certificate
```

4.

파일을 저장합니다.

5.

플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.certificate/README.md` 파일을 참조하십시오.
- `ansible-playbook(1)` 도움말 페이지를 참조하십시오.

15.3. 인증서 문제 및 갱신 시스템 역할을 사용하여 IDM CA에서 새 인증서 요청

인증 문제 및 업데이트 시스템 역할을 통해 통합 인증 기관(CA)과 함께 IdM 서버를 사용하는 동안 **anible-core** 를 사용하여 인증서를 발행할 수 있습니다. 따라서 IdM을 CA로 사용할 때 여러 시스템의 인증서 신뢰 체인을 효율적이고 일관되게 관리할 수 있습니다.

이 프로세스에서는 **certmonger** 공급자를 사용하고 **getcert** 명령을 통해 인증서를 요청합니다.



참고

기본적으로 **certmonger** 는 만료되기 전에 인증서를 자동으로 업데이트하려고 합니다. **Ansible** 플레이북의 **auto_renew** 매개변수를 **no** 로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#) 를 참조하십시오.

절차

1. **선택 사항:** 인벤토리 파일(예: **inventory.file**)을 생성합니다.

```
$ touch inventory.file
```

2. 인벤토리 파일을 열고 인증서를 요청할 호스트를 정의합니다. 예를 들면 다음과 같습니다.

```
[webserver]
server.idm.example.com
```

3.

플레이북 파일을 만듭니다(예: `request-certificate.yml`):

- 웹 서버와 같이 인증서를 요청할 호스트를 포함하도록 호스트를 설정합니다.
- 다음을 포함하도록 `certificate_requests` 변수를 설정합니다.
 - `name` 매개변수를 `mycert` 와 같은 인증서의 원하는 이름으로 설정합니다.
 - `dns` 매개변수를 인증서에 포함할 도메인(예: `www.example.com`)으로 설정합니다.
 - `principal` 매개 변수를 설정하여 `HTTP/www.example.com@EXAMPLE.COM` 와 같은 `Kerberos` 주체를 지정합니다.
 - `ca` 매개변수를 `ipa` 로 설정합니다.
- 역할에 `rhel-system-roles.certificate` 역할을 설정합니다.

이 예제는 플레이북 파일입니다.

```
---
- hosts: webserver
  vars:
    certificate_requests:
      - name: mycert
        dns: www.example.com
        principal: HTTP/www.example.com@EXAMPLE.COM
        ca: ipa

  roles:
    - rhel-system-roles.certificate
```

4.

파일을 저장합니다.

5. 플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.certificate/README.md](#) 파일을 참조하십시오.
- [ansible-playbook\(1\)](#) 도움말 페이지를 참조하십시오.

15.4. 인증서 문제 및 갱신 시스템 역할을 사용하여 인증서 발급 후 실행할 명령 지정

인증서 시스템 문제 및 업데이트 역할을 사용하면 인증서가 발행되거나 갱신되기 전과 후에 **Ansible Core**를 사용하여 명령을 실행할 수 있습니다.

다음 예에서 관리자는 **www.example.com**의 자체 서명 인증서 전에 **httpd** 서비스를 중지한 후 나중에 다시 시작합니다.



참고

기본적으로 **certmonger**는 만료되기 전에 인증서를 자동으로 업데이트하려고 합니다. **Ansible** 플레이북의 **auto_renew** 매개변수를 **no**로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#)를 참조하십시오.

절차

1.

선택 사항: 인벤토리 파일(예: **inventory.file**)을 생성합니다.

```
$ touch inventory.file
```

2.

인벤토리 파일을 열고 인증서를 요청할 호스트를 정의합니다. 예를 들면 다음과 같습니다.

```
[webserver]
server.idm.example.com
```

3.

플레이북 파일을 만듭니다(예: **request-certificate.yml**):

- 웹 서버와 같이 인증서를 요청할 호스트를 포함하도록 호스트를 설정합니다.
- 다음을 포함하도록 **certificate_requests** 변수를 설정합니다.
 - **name** 매개변수를 **mycert** 와 같은 인증서의 원하는 이름으로 설정합니다.
 - **dns** 매개변수를 인증서에 포함할 도메인(예: **www.example.com**)으로 설정합니다.
 - **self-sign** 과 같은 인증서를 발급하는 데 사용할 **ca** 매개변수를 **CA**로 설정합니다.
 - 이 인증서를 발행하거나 갱신하기 전에 실행할 명령에 **run_before** 매개 변수를 **systemctl stop httpd.service** 와 같이 설정합니다.
 - 이 인증서가 발행되거나 갱신된 후 실행할 명령에 **run_after** 매개 변수를 다음과 같이 설정합니다(예: **systemctl start httpd.service**).
- 역할에 **rhel-system-roles.certificate** 역할을 설정합니다.

이 예제는 플레이북 파일입니다.

```

- hosts: webserver
vars:
  certificate_requests:
    - name: mycert
      dns: www.example.com
      ca: self-sign
      run_before: systemctl stop httpd.service
      run_after: systemctl start httpd.service

roles:
  - rhel-system-roles.certificate

```

4.

파일을 저장합니다.

5.

플레이북을 실행합니다.

```
$ ansible-playbook -i inventory.file request-certificate.yml
```

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.certificate/README.md](#) 파일을 참조하십시오.
- [ansible-playbook\(1\)](#) 도움말 페이지를 참조하십시오.

16장. RHEL 시스템 역할을 사용하여 KDUMP 구성

Ansible을 사용하여 **kdump**를 관리하려면 **RHEL 8**에서 사용할 수 있는 **RHEL 시스템 역할** 중 하나인 **커널 덤프 역할**을 사용할 수 있습니다.

커널 덤프 역할을 사용하면 나중에 분석을 위해 시스템 메모리의 내용을 저장할 위치를 지정할 수 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 소개](#)를 참조하십시오.

16.1. 커널 덤프 RHEL 시스템 역할

커널 덤프 시스템 역할을 사용하면 여러 시스템에서 기본 커널 덤프 매개변수를 설정할 수 있습니다.

16.2. 커널 덤프 역할 매개변수

커널 덤프의 **RHEL 시스템 역할**에 사용되는 매개변수는 다음과 같습니다.

역할 변수	설명
kdump_path	vmcore 가 기록되는 경로입니다. kdump_target 이 null이 아닌 경우 경로는 해당 덤프 대상을 기준으로 합니다. 그렇지 않으면 루트 파일 시스템의 절대 경로여야 합니다.

추가 리소스

- [makedumpfile\(8\) 도움말 페이지](#).
- **kdump**에 사용되는 매개변수 및 커널 **Dumps** 시스템 역할에 대한 자세한 내용은 [/usr/share/ansible/roles/rhel-system-roles.tlog/README.md](#) 파일을 참조하십시오.

16.3. RHEL 시스템 역할을 사용하여 KDUMP 구성

Ansible 플레이북을 실행하여 커널 덤프 시스템 역할을 사용하여 여러 시스템에서 기본 커널 덤프 매개변수를 설정할 수 있습니다.



주의

커널 덤프스 역할은 **/etc/kdump.conf** 파일을 교체하여 관리 호스트의 **kdump** 설정을 완전히 대체합니다. 또한 커널 **Dumps** 역할이 적용되는 경우 **/etc/sysconfig/kdump** 파일을 대체하여 역할 변수에 지정하지 않은 경우에도 이전 **kdump** 설정도 모두 교체됩니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **kdump**를 배포하려는 시스템을 나열하는 인벤토리 파일이 있습니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: kdump-test
  vars:
    kdump_path: /var/crash
  roles:
    - rhel-system-roles.kdump
```

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

추가 리소스

- 커널 덤프 역할 변수에 대한 자세한 내용은 `/usr/share/doc/rhel-system-roles/kdump` 디렉터리의 **README.md** 또는 **README.html** 파일을 참조하십시오.
- [시스템 역할 적용을](#) 참조하십시오.
- **rhel-system-roles** 패키지 `/usr/share/ansible/roles/rhel-system-roles.kdump/README.html`과 함께 설치된 설명서

17장. RHEL 시스템 역할을 사용하여 로컬 스토리지 관리

Ansible을 사용하여 **LVM** 및 로컬 파일 시스템(**FS**)을 관리하려면 **RHEL 9**에서 사용할 수 있는 **RHEL** 시스템 역할 중 하나인 **Storage** 역할을 사용할 수 있습니다.

Storage 역할을 사용하면 여러 시스템에서 디스크 및 논리 볼륨에서 파일 시스템 관리를 자동화할 수 있으며 **RHEL 7.7**부터 시작하는 모든 **RHEL** 버전에서 사용할 수 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 소개](#)를 참조하십시오.

17.1. 스토리지 역할 소개

스토리지 역할은 다음을 관리할 수 있습니다.

- 분할되지 않은 디스크의 파일 시스템
- 논리 볼륨 및 파일 시스템을 포함하여 **LVM** 볼륨 그룹 완료

Storage 역할을 사용하면 다음 작업을 수행할 수 있습니다.

- 파일 시스템 생성
- 파일 시스템 제거
- 파일 시스템 마운트
- 파일 시스템 마운트 해제
- **LVM** 볼륨 그룹 만들기

- **LVM 볼륨 그룹 제거**
- 논리 볼륨 생성
- 논리 볼륨 제거
- **RAID 볼륨 생성**
- **RAID 볼륨 제거**
- **RAID를 사용하여 LVM 풀 생성**
- **RAID를 사용하여 LVM 풀 제거**

17.2. 스토리지 시스템 역할에서 스토리지 장치를 식별하는 매개변수

스토리지 역할 구성은 다음 변수에서 나열하는 파일 시스템, 볼륨 및 풀에만 영향을 미칩니다.

storage_volumes

관리할 파티션되지 않은 모든 디스크의 파일 시스템 목록입니다.

파티션은 현재 지원되지 않습니다.

storage_pools

관리할 풀 목록입니다.

현재 지원되는 유일한 풀 유형은 **LVM**입니다. **LVM**을 사용하면 풀은 볼륨 그룹(**VG**)을 나타냅니다. 각 풀에는 역할에서 관리할 볼륨 목록이 있습니다. **LVM**에서 각 볼륨은 파일 시스템의 논리 볼륨(**LV**)에 해당합니다.

17.3. 블록 장치에 **XFS** 파일 시스템을 생성하는 **ANSIBLE** 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 기본 매개변수를 사용하여 블록 장치에 **XFS** 파일 시스템을 생성하는 **Storage** 역할을 적용합니다.



주의

스토리지 역할은 파티션되지 않은 전체 디스크 또는 논리 볼륨(LV)에서만 파일 시스템을 생성할 수 있습니다. 파티션에 파일 시스템을 만들 수 없습니다.

예 17.1. /dev/sdb에 XFS를 생성하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
  roles:
    - rhel-system-roles.storage
```

- 볼륨 이름(예의 *베어 메탈*)은 현재 임의적입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.
- **XFS**는 **RHEL 9**의 기본 파일 시스템이므로 **fs_type: xfs** 행을 생략할 수 있습니다.
- **LV**에 파일 시스템을 만들려면 포함된 볼륨 그룹을 포함하여 **disks:** 속성 아래에 **LVM** 설정을 제공하십시오.

LV 장치에 대한 경로를 제공하지 마십시오.

추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 파일.

17.4. ANSIBLE 플레이북의 예제로 파일 시스템을 지속적으로 마운트합니다.

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **Storage** 역할을 적용하여 **XFS** 파일 시스템을 즉시 영구적으로 마운트합니다.

예 17.2. /dev/sdb에 파일 시스템을 /mnt/data에 마운트하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- 이 플레이북은 파일 시스템을 **/etc/fstab** 파일에 추가하고 파일 시스템을 즉시 마운트합니다.
- **/dev/sdb** 장치 또는 마운트 지점 디렉터리의 파일 시스템이 없으면 플레이북에서 해당 장치를 생성합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

17.5. 논리 볼륨을 관리하는 ANSIBLE 플레이북 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 역할을 적용하여 볼륨 그룹에 **LVM** 논리 볼륨을 만듭니다.

예 17.3. myvg 볼륨 그룹에 mylv 논리 볼륨을 생성하는 플레이북

```
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - sda
```

```

- sdb
- sdc
volumes:
- name: mylv
  size: 2G
  fs_type: ext4
  mount_point: /mnt
roles:
- rhel-system-roles.storage

```

- **myvg** 볼륨 그룹은 다음 디스크로 구성됩니다.
 - **/dev/sda**
 - **/dev/sdb**
 - **/dev/sdc**
- **myvg** 볼륨 그룹이 이미 있는 경우 플레이북은 논리 볼륨을 볼륨 그룹에 추가합니다.
- **myvg** 볼륨 그룹이 없는 경우 플레이북에서 이 그룹을 생성합니다.
- 이 플레이북은 **mylv** 논리 볼륨에 **Ext4** 파일 시스템을 생성하고 **/mnt**에 파일 시스템을 영구적으로 마운트합니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

17.6. 온라인 블록 삭제 활성화를 위한 예제 **ANSIBLE** 플레이북

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 온라인 블록 삭제가 활성화된 **XFS** 파일 시스템을 마운트하는 **Storage** 역할을 적용합니다.

예 17.4. **/mnt/data/**에서 온라인 블록 삭제 기능을 활성화하는 플레이북

```

---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: xfs
        mount_point: /mnt/data
        mount_options: discard
  roles:
    - rhel-system-roles.storage

```

추가 리소스

- [Ansible 플레이북의 예제로 파일 시스템을 지속적으로 마운트합니다.](#)
- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) 파일.

17.7. EXT4 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **Ext4** 파일 시스템을 만들고 마운트하기 위해 **Storage** 역할을 적용합니다.

예 17.5. /dev/sdb에 Ext4를 생성하고 이를 /mnt/data에 마운트하는 플레이북입니다.

```

---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext4
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage

```

- 플레이북은 **/dev/sdb** 디스크에 파일 시스템을 생성합니다.

- 플레이북은 **/mnt/data** 디렉터리에 파일 시스템을 영구적으로 마운트합니다.
- 파일 시스템의 레이블은 **label-name**입니다.

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

17.8. EXT3 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **Ext3** 파일 시스템을 만들고 마운트하기 위해 **Storage** 역할을 적용합니다.

예 17.6. **/dev/sdb**에 **Ext3**를 생성하고 이를 **/mnt/data**에 마운트하는 플레이북입니다.

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - sdb
        fs_type: ext3
        fs_label: label-name
        mount_point: /mnt/data
  roles:
    - rhel-system-roles.storage
```

- 플레이북은 **/dev/sdb** 디스크에 파일 시스템을 생성합니다.
- 플레이북은 **/mnt/data** 디렉터리에 파일 시스템을 영구적으로 마운트합니다.
- 파일 시스템의 레이블은 **label-name**입니다.

추가 리소스

-

`/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 파일.

17.9. 스토리지 RHEL 시스템 역할을 사용하여 기존 EXT4 또는 EXT3 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 **Playbook**은 블록 장치의 기존 **Ext4** 또는 **Ext3** 파일 시스템의 크기를 조정하는 **Storage** 역할을 적용합니다.

예 17.7. 디스크에 단일 볼륨을 설정하는 플레이북

```
---
- name: Create a disk device mounted on /opt/barefs
  hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
  size: 12 GiB
  fs_type: ext4
  mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

-

이전 예제의 볼륨이 이미 존재하는 경우 볼륨의 크기를 조정하려면 매개 변수 크기에 다른 값으로 동일한 플레이북을 실행해야 합니다. 예를 들면 다음과 같습니다.

예 17.8. /dev/sdb에서 ext4 의 크기를 조정하는 플레이북

```
---
- name: Create a disk device mounted on /opt/barefs
  hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
        disks:
          - /dev/sdb
  size: 10 GiB
  fs_type: ext4
  mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

-

볼륨 이름(예: 바프)은 현재 임의적입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디

스크 장치에서 볼륨을 식별합니다.



참고

다른 파일 시스템에서 **Resizing** 작업을 사용하면 작업 중인 장치의 데이터를 삭제할 수 있습니다.

추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 파일.

17.10. 스토리지 RHEL 시스템 역할을 사용하여 LVM의 기존 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 **RHEL** 시스템 역할을 적용하여 파일 시스템으로 **LVM** 논리 볼륨의 크기를 조정합니다.



주의

다른 파일 시스템에서 **Resizing** 작업을 사용하면 작업 중인 장치의 데이터를 삭제할 수 있습니다.

예 17.9. myvg 볼륨 그룹의 기존 mylv1 및 mylv2 논리 볼륨의 크기를 조정하는 플레이북

```
---
- hosts: all
  vars:
    storage_pools:
      - name: myvg
        disks:
          - /dev/sda
          - /dev/sdb
          - /dev/sdc
        volumes:
          - name: mylv1
            size: 10 GiB
            fs_type: ext4
```

```

    mount_point: /opt/mount1
  - name: mylv2
    size: 50 GiB
    fs_type: ext4
    mount_point: /opt/mount2

- name: Create LVM pool over three disks
  include_role:
    name: rhel-system-roles.storage

```

- 이 플레이북은 다음과 같은 기존 파일 시스템의 크기를 조정합니다.
 - /opt/mount1 에 마운트된 mylv1 볼륨의 Ext4 파일 시스템은 10GiB로 크기를 조정합니다.
 - /opt/mount2 에 마운트된 mylv2 볼륨의 Ext4 파일 시스템은 50GiB로 크기를 조정합니다.

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) 파일.

17.11. 스토리지 RHEL 시스템 역할을 사용하여 스왑 파티션을 생성하는 ANSIBLE 플레이북 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 **Storage** 역할을 적용하여 스왑 파티션을 생성하거나, 스왑 파티션이 없는 경우, 이미 존재하는 경우 기본 매개 변수를 사용하여 블록 장치에 수정합니다.

예 17.10. /dev/sdb에서 기존 XFS를 생성하거나 수정하는 플레이북

```

---
- name: Create a disk device with swap
- hosts: all
vars:
  storage_volumes:
    - name: swap_fs
      type: disk
      disks:
        - /dev/sdb
size: 15 GiB
fs_type: swap
roles:
  - rhel-system-roles.storage

```

- 볼륨 이름(예의 **swap_fs**)은 현재 임의적입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.

추가 리소스

- `/usr/share/ansible/roles/rhel-system-roles.storage/README.md` 파일.

17.12. 스토리지 시스템 역할을 사용하여 RAID 볼륨 구성

스토리지 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**에서 **RAID** 볼륨을 구성할 수 있습니다. 이 섹션에서는 요구 사항에 맞게 **RAID** 볼륨을 구성하기 위해 사용할 수 있는 매개 변수를 사용하여 **Ansible** 플레이북을 설정하는 방법을 알아봅니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- 스토리지 시스템 역할을 사용하여 **RAID** 볼륨을 배포하려는 시스템을 자세히 설명하는 인벤토리 파일이 있습니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- hosts: all
  vars:
    storage_safe_mode: false
    storage_volumes:
      - name: data
        type: raid
        disks: [sdd, sde, sdf, sdg]
        raid_level: raid0
        raid_chunk_size: 32 KiB
        mount_point: /mnt/data
        state: present
  roles:
    - name: rhel-system-roles.storage
```



주의

장치 이름은 특정 상황에서 변경될 수 있습니다(예: 시스템에 새 디스크를 추가할 때). 따라서 데이터 손실을 방지하기 위해 플레이북에서 특정 디스크 이름을 사용하지 않는 것이 좋습니다.

2.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

- [/usr/share/ansible/roles/rhel-system-roles.storage/README.md](#) 파일.

17.13. 스토리지 시스템 역할을 사용하여 RAID로 LVM 풀 구성

스토리지 시스템 역할을 사용하면 Red Hat Ansible Automation Platform을 사용하여 RHEL에서 RAID를 사용하여 LVM 풀을 구성할 수 있습니다. 이 섹션에서는 RAID로 LVM 풀을 구성하기 위해 사용할 수 있는 매개 변수를 사용하여 Ansible 플레이북을 설정하는 방법을 배웁니다.

사전 요구 사항

- Ansible Core 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 rhel-system-roles 패키지가 설치되어 있습니다.
- 스토리지 시스템 역할을 사용하여 RAID로 LVM 풀을 구성할 시스템을 자세히 설명하는 인벤토리 파일이 있습니다.

설치

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- hosts: all
  vars:
    storage_safe_mode: false
    storage_pools:
      - name: my_pool
        type: lvm
        disks: [sdh, sdi]
        raid_level: raid1
        volumes:
          - name: my_pool
            size: "1 GiB"
            mount_point: "/mnt/app/shared"
            fs_type: xfs
            state: present
  roles:
    - name: rhel-system-roles.storage
```



참고

RAID를 사용하여 **LVM** 풀을 생성하려면 **raid_level** 매개변수를 사용하여 **RAID** 유형을 지정해야 합니다.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일.

17.14. 스토리지 RHEL 시스템 역할을 사용하여 LVM에서 VDO 볼륨을 압축하고 중복하기 위한 ANSIBLE 플레이북 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 플레이북은 스토리지 **RHEL** 시스템 역할을 적

용하여 **VDO**(가상 데이터 최적화 도구) 볼륨을 사용하여 압축 및 중복 제거를 **LVM(Logical Manager 볼륨)**에 적용할 수 있습니다.

예 17.11. myvg 볼륨 그룹에 mylv1 LVM VDO 볼륨을 생성하는 플레이북

```
---
- name: Create LVM VDO volume under volume group 'myvg'
  hosts: all
  roles:
    - rhel-system-roles.storage
  vars:
    storage_pools:
      - name: myvg
        disks:
          - /dev/sdb
        volumes:
          - name: mylv1
            compression: true
            deduplication: true
            vdo_pool_size: 10 GiB
            size: 30 GiB
            mount_point: /mnt/app/shared
```

이 예에서 압축 및 중복 제거 풀이 **true**로 설정되어 **VDO**가 사용되도록 지정합니다. 다음은 이러한 매개 변수의 사용법을 설명합니다.

- 중복 제거 는 스토리지 볼륨에 저장된 중복된 데이터를 복제하는 데 사용됩니다.
- 압축은 스토리지 볼륨에 저장된 데이터를 압축하는 데 사용되며 스토리지 용량이 늘어납니다.
- **EgressIP_pool_size**는 볼륨에서 장치에서 가져오는 실제 크기를 지정합니다. **VDO** 볼륨의 가상 크기는 **size** 매개 변수로 설정됩니다. 알림: **LVM VDO**의 스토리지 역할 때문에 풀당 하나의 볼륨만 압축 및 중복 제거를 사용할 수 있습니다.

17.15. 스토리지 시스템 역할을 사용하여 LUKS 암호화된 볼륨 생성

Storage 역할을 사용하여 **Ansible** 플레이북을 실행하여 **LUKS**로 암호화된 볼륨을 생성하고 구성할 수 있습니다.

사전 요구 사항

-

Crypto Policies System Role으로 구성하려는 시스템인 하나 이상의 **관리형 노드**에 대한 액세스 및 사용 권한.

- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 제어 노드에 대한 액세스 및 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.

중요

RHEL 8.0-8.5는 **Ansible** 기반 자동화를 위해 **Ansible Engine 2.9**가 포함된 별도의 **Ansible** 리포지토리에 대한 액세스를 제공했습니다. **Ansible Engine**에는 **ansible**, **ansible-playbook**, **docker** 및 **podman** 과 같은 커넥터, 여러 플러그인 및 모듈과 같은 명령줄 유틸리티가 포함되어 있습니다. **Ansible Engine**을 확보하고 설치하는 방법에 대한 자세한 내용은 [Red Hat Ansible Engine 지식베이스를 다운로드하고 설치하는 방법](#) 문서를 참조하십시오.

RHEL 8.6 및 **9.0**에서는 **Ansible** 명령줄 유틸리티, 명령 및 소규모의 기본 제공 **Ansible** 플러그인 세트가 포함된 **Ansible Core**(**ansible-core** 패키지로 제공)를 도입했습니다. **RHEL**은 **AppStream** 리포지토리를 통해 이 패키지를 제공하며 제한된 지원 범위를 제공합니다. 자세한 내용은 [RHEL 9 및 RHEL 8.6 이상 AppStream 리포지토리 지식 베이스에 포함된 Ansible Core 패키지에 대한 지원 범위](#)를 참조하십시오.

- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
- hosts: all
vars:
  storage_volumes:
    - name: barefs
      type: disk
  disks:
    - sdb
  fs_type: xfs
  fs_label: label-name
  mount_point: /mnt/data
```

```

    encryption: true
    encryption_password: your-password
  roles:
    - rhel-system-roles.storage

```

2.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory.file /path/to/file/playbook.yml
```

추가 리소스

•

/usr/share/ansible/roles/rhel-system-roles.storage/README.md 파일

17.16. 스토리지 RHEL 시스템 역할을 사용하여 풀 볼륨 크기를 백분율로 표시하는 **ANSIBLE** 플레이북의 예

이 섹션에서는 예제 **Ansible** 플레이북을 제공합니다. 이 **Playbook**은 스토리지 시스템 역할을 적용하여 풀의 총 크기의 백분율로 **LVM(Logical Manager 볼륨)** 볼륨 크기를 표시할 수 있도록 합니다.

예 17.12. 볼륨 크기를 풀의 총 크기의 백분율로 표시하는 플레이북

```

---
- name: Express volume sizes as a percentage of the pool's total size
  hosts: all
  roles:
    - rhel-system-roles.storage
  vars:
    storage_pools:
      - name: myvg
        disks:
          - /dev/sdb
        volumes:
          - name: data
            size: 60%
            mount_point: /opt/mount/data
          - name: web
            size: 30%
            mount_point: /opt/mount/web
          - name: cache
            size: 10%
            mount_point: /opt/cache/mount

```

이 예제에서는 **LVM** 볼륨의 크기를 풀 크기의 백분율로 지정합니다. 예를 들면 다음과 같습니다. **"60%"**. 또한 **LVM** 볼륨의 크기를 사람이 읽을 수 있는 파일 시스템의 크기(예: **"10g"** 또는 **"50GiB"**)에서 풀 크기로 지정할 수도 있습니다.

17.17. 추가 리소스

- [/usr/share/doc/rhel-system-roles/storage/](#)
- [/usr/share/ansible/roles/rhel-system-roles.storage/](#)

18장. RHEL 시스템 역할을 사용하여 시간 동기화 구성

시간 동기화 **RHEL** 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**의 여러 대상 시스템에서 시간 동기화를 관리할 수 있습니다.

18.1. 시간 동기화 시스템 역할

시간 동기화 **RHEL** 시스템 역할을 사용하여 여러 대상 시스템에서 시간 동기화를 관리할 수 있습니다.

시간 동기화 역할은 **NTP** 서버 또는 **PTP** 도메인에서 시스템 클럭을 동기화하기 위해 **NTP** 클라이언트 또는 **PTP** 복제본으로 작동하도록 **NTP** 또는 **PTP** 구현을 설치하고 구성합니다.

시간 동기화 역할을 사용하면 시스템에서 **NTP** 프로토콜을 구현하는 데 **ntp** 또는 **chrony**를 사용하는지 여부와 관계없이 **RHEL 6**부터 시작하는 모든 **Red Hat Enterprise Linux** 버전에서 동일한 플레이북을 사용할 수 있으므로 **chrony** 로 쉽게 마이그레이션할 수도 있습니다.

18.2. 단일 서버 풀에 대한 시간 동기화 시스템 역할 적용

다음 예제에서는 하나의 서버 풀이 있는 상황에서 시간 동기화 역할을 적용하는 방법을 보여줍니다.



주의

시간 동기화 역할은 관리 호스트에서 지정된 또는 탐지된 공급자 서비스의 구성을 대체합니다. 이전 설정은 역할 변수에 지정되지 않은 경우에도 손실됩니다. **timesync_ntp_provider** 변수가 정의되지 않은 경우 보존된 유일한 설정은 공급자 선택입니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

- 시간 동기화 시스템 역할을 배포하려는 시스템을 나열하는 인벤토리 파일이 있습니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: 2.rhel.pool.ntp.org
        pool: yes
        iburst: yes
  roles:
    - rhel-system-roles.timesync
```

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

18.3. 클라이언트 서버에 시간 동기화 시스템 역할 적용

시간 동기화 역할을 사용하여 **NTP** 클라이언트에서 네트워크 시간 보안(**NTS**)을 활성화할 수 있습니다. **Network Time Security(NTS)**는 **NTP(Network Time Protocol)**에 지정된 인증 메커니즘입니다. 서버와 클라이언트 간에 교환된 **NTP** 패킷이 변경되지 않았는지 확인합니다.



주의

시간 동기화 역할은 관리 호스트에서 지정된 또는 탐지된 공급자 서비스의 구성을 대체합니다. 이전 설정은 역할 변수에 지정되지 않은 경우에도 손실됩니다. **timesync_ntp_provider** 변수가 정의되지 않은 경우 보존된 유일한 설정은 공급자 선택입니다.

사전 요구 사항

- **timesync** 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다.
- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- 시간 동기화 시스템 역할을 배포하려는 시스템을 나열하는 인벤토리 파일이 있습니다.
- **chrony** NTP 공급자 버전은 **4.0** 이상입니다.

절차

1. 다음 콘텐츠를 사용하여 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: timesync-test
  vars:
    timesync_ntp_servers:
      - hostname: ptbtime1.ptb.de
        iburst: yes
        nts: yes
  roles:
    - rhel-system-roles.timesync
```

ptbtime1.ptb.de 는 공용 서버의 예입니다. 다른 공용 서버 또는 자체 서버를 사용하려는 경우도 있습니다.

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

검증

1.

클라이언트 머신에서 테스트를 수행합니다.

chronyc -N authdata

Name/IP address Mode KeyID Type KLen Last Atmp NAK Cook CLen

=====

ptbtime1.ptb.de NTS 1 15 256 157 0 0 8 100

2.

보고된 쿠키 수가 0보다 큰지 확인합니다.

추가 리소스

•

chrony.conf(5) man page

18.4. 시간 동기화 시스템 역할 변수

다음 변수를 시간 동기화 역할에 전달할 수 있습니다.

•

timesync_ntp_servers:

역할 변수 설정	설명
hostname: host.example.com	서버의 호스트 이름 또는 주소
minpoll: 번호	최소 폴링 간격입니다. 기본값: 6
maxpoll: 번호	최대 폴링 간격입니다. 기본값: 10
보이지 않음: 예	빠른 초기 동기화를 활성화하는 플래그입니다. 기본값: no
풀: yes	호스트 이름의 각 주소가 별도의 NTP 서버임을 나타내는 플래그입니다. 기본값: no
NTS: yes	네트워크 시간 보안(NTS)을 활성화하도록 플래그입니다. 기본값: no. chrony >= 4.0에서만 지원됩니다.

추가 리소스

•

시간 동기화 역할 변수에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고

/usr/share/doc/rhel-system-roles/timesync 디렉터리에서 **README.md** 또는 **README.html** 파일을 참조하십시오.

19장. RHEL 시스템 역할을 사용하여 성능 모니터링

시스템 관리자는 **Metrics RHEL** 시스템 역할을 사용하여 시스템의 성능을 모니터링할 수 있습니다.

19.1. 지표 시스템 역할 소개

RHEL 시스템 역할은 여러 **RHEL** 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 **Ansible** 역할 및 모듈의 컬렉션입니다. **Metrics** 시스템 역할은 로컬 시스템에 대한 성능 분석 서비스를 구성하고, 선택적으로 로컬 시스템에서 모니터링할 원격 시스템 목록을 포함합니다. 지표 시스템 역할을 사용하면 **pcp** 를 개별적으로 구성하지 않고도 **pcp**를 사용하여 플레이북에서 설정 및 배포를 처리하므로 **pcp** 를 사용하여 시스템 성능을 모니터링할 수 있습니다.

표 19.1. 지표 시스템 역할 변수

역할 변수	설명	사용 예
metrics_monitored_hosts	대상 호스트에서 분석할 원격 호스트 목록입니다. 이러한 호스트에는 대상 호스트에 기록된 지표가 있으므로 각 호스트에 대해 /var/log 아래에 디스크 공간이 충분히 있는지 확인하십시오.	metrics_monitored_hosts: ["webserver.example.com", "database.example.com"]
metrics_retention_days	삭제하기 전에 성능 데이터 보존을 위한 일 수를 구성합니다.	metrics_retention_days: 14
metrics_graph_service	pcp 및 grafana 를 통해 성능 데이터 시각화를 위해 서비스를 사용하여 호스트를 설정할 수 있는 부울 플래그입니다. 기본적으로 false 로 설정합니다.	metrics_graph_service: no
metrics_query_service	시계열 쿼리 서비스에서 빨간색을 통해 기록된 pcp 지표를 쿼리할 수 있도록 호스트를 설정할 수 있는 부울 플래그입니다. 기본적으로 false 로 설정합니다.	metrics_query_service: no
metrics_provider	메트릭을 제공하는 데 사용할 메트릭 수집기를 지정합니다. 현재 pcp 는 지원되는 유일한 지표 공급자입니다.	metrics_provider: "pcp"



참고

metrics_connections에 사용되는 매개 변수 및 지표 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.metrics/README.md** 파일을 참조하십시오.

19.2. 시각화로 로컬 시스템을 모니터링하기 위해 메트릭 시스템 역할을 사용

이 절차에서는 **Grafana**를 통해 데이터 시각화를 동시에 프로비저닝하는 동안 **Metrics RHEL** 시스템 역할을 사용하여 로컬 시스템을 모니터링하는 방법을 설명합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 모니터링할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1. 인벤토리에 다음 콘텐츠를 추가하여 **/etc/ansible/hosts** Ansible 인벤토리에서 **localhost**를 구성합니다.

```
localhost ansible_connection=local
```

2. 다음 콘텐츠를 사용하여 **Ansible** 플레이북을 생성합니다.

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
  roles:
    - rhel-system-roles.metrics
```

3. **Ansible Playbook**을 실행합니다.

```
# ansible-playbook name_of_your_playbook.yml
```



참고

metrics_graph_service 부울이 **value="yes"**로 설정되어 있으므로 **Grafana**는 데이터 소스로 **pcp**를 추가하여 자동으로 설치되고 프로비저닝됩니다.

4.

시스템에서 수집되는 지표의 시각화를 보려면 **Grafana 웹 UI 액세스에 설명된 대로 grafana 웹 인터페이스에 액세스합니다.**

19.3. 지표 시스템 역할을 사용하여 자신을 모니터링하기 위해 개별 시스템 세트를 설정

이 절차에서는 메트릭을 사용하여 모니터링할 시스템 제품군을 설정하는 데 **Metrics** 시스템 역할을 사용하는 방법을 설명합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **SSH** 연결이 설정되어 있습니다.

절차

1.

플레이북을 통해 모니터링할 머신의 이름 또는 IP를 대괄호로 묶은 식별 그룹 이름 아래에 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.

```
[remotes]
webserver.example.com
database.example.com
```

2.

다음 콘텐츠를 사용하여 **Ansible** 플레이북을 생성합니다.

```
---
- hosts: remotes
  vars:
    metrics_retention_days: 0
  roles:
    - rhel-system-roles.metrics
```

3.

Ansible Playbook을 실행합니다.

```
# ansible-playbook name_of_your_playbook.yml -k
```

여기서 **-k** 는 원격 시스템에 연결하기 위한 암호를 묻습니다.

19.4. 메트릭 시스템 역할을 사용하여 로컬 시스템을 통해 중앙 집중식으로 시스템 그룹을 모니터링할 수 있습니다.

이 절차에서는 **Metrics** 시스템 역할을 사용하여 시스템을 중앙에서 모니터링하도록 로컬 머신을 설정하는 방법을 설명하고, **grafana** 를 통해 데이터의 시각화를 프로비저닝하고 **redis** 를 통해 데이터를 쿼리합니다.

사전 요구 사항

•

Ansible Core 패키지는 제어 시스템에 설치됩니다.

•

플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1.

다음 콘텐츠를 사용하여 **Ansible** 플레이북을 생성합니다.

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
    metrics_query_service: yes
    metrics_retention_days: 10
    metrics_monitored_hosts: ["database.example.com", "webserver.example.com"]
  roles:
    - rhel-system-roles.metrics
```

2.

Ansible Playbook을 실행합니다.

```
# ansible-playbook name_of_your_playbook.yml
```



참고

metrics_graph_service 부울 및 **metrics_query_service** 부울이 **value="yes"**로 설정되므로 **grafana** 는 자동으로 **pcp** 데이터 레코딩을 사용하여 **redis** 로 인덱싱된 데이터 소스로 추가되어 **pcp** 쿼리 언어를 사용하여 데이터를 쉽게 쿼리할 수 있습니다.

3.

시스템에서 중앙 집중적으로 수집하고 데이터를 쿼리하려면 **Grafana 웹 UI**에 액세스하는 데 설명된 대로 **grafana** 웹 인터페이스에 액세스합니다.

19.5. METRICS 시스템 역할을 사용하여 시스템을 모니터링하는 동안 인증 설정

PCP는 SASL(Simple Authentication Security Layer) 프레임워크를 통해 **scram-sha-256** 인증 메커니즘을 지원합니다. **Metrics RHEL** 시스템 역할은 **scram-sha-256** 인증 메커니즘을 사용하여 인증을 설정하는 단계를 자동화합니다. 다음 절차에서는 **Metrics RHEL** 시스템 역할을 사용하여 인증을 설정하는 방법을 설명합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1.

다음에 대한 인증을 설정하려는 **Ansible** 플레이북에 다음 변수를 포함합니다.

```
---
vars:
  metrics_username: your_username
  metrics_password: your_password
```

2.

Ansible Playbook을 실행합니다.

```
# ansible-playbook name_of_your_playbook.yml
```

검증 단계

- **sasl** 구성을 확인합니다.

```
# pminfo -f -h "pcp://ip_adress?username=your_username" disk.dev.read
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

ip_adress 는 호스트의 IP 주소로 교체해야 합니다.

19.6. METRICS 시스템 역할을 사용하여 SQL SERVER에 대한 메트릭 컬렉션을 구성하고 활성화합니다.

이 절차에서는 **Metrics RHEL** 시스템 역할을 사용하여 로컬 시스템의 **pcp** 를 통해 **Microsoft SQL Server**에 대한 메트릭 컬렉션 및 구성을 자동화하는 방법을 설명합니다.

사전 요구 사항

- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- 모니터링할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **Red Hat Enterprise Linux**용 **Microsoft SQL Server**를 설치하고 **SQL** 서버에 대한 '신뢰할 수 있는' 연결을 설정했습니다. [SQL Server 설치를 참조하고 Red Hat에 데이터베이스를 만듭니다.](#)
- **Red Hat Enterprise Linux**용 **Microsoft rsh** 드라이버를 설치했습니다. [Red Hat Enterprise Server 및 Oracle Linux](#) 를 참조하십시오.

절차

1. 인벤토리에 다음 콘텐츠를 추가하여 **/etc/ansible/hosts** **Ansible** 인벤토리에서 **localhost** 를 구성합니다.

```
localhost ansible_connection=local
```

2. 다음 콘텐츠가 포함된 **Ansible** 플레이북을 생성합니다.

```
---
```

```
- hosts: localhost
roles:
  - role: rhel-system-roles.metrics
vars:
  metrics_from_mssql: yes
```

3.

Ansible Playbook을 실행합니다.

```
# ansible-playbook name_of_your_playbook.yaml
```

검증 단계

- **pcp** 명령을 사용하여 **SQL Server PMDA 에이전트(mssql)**가 로드되고 실행 중인지 확인합니다.

```
# pcp
platform: Linux rhel82-2.local 4.18.0-167.el8.x86_64 #1 SMP Sun Dec 15 01:24:23 UTC
2019 x86_64
hardware: 2 cpus, 1 disk, 1 node, 2770MB RAM
timezone: PDT+7
services: pmcd pmproxy
  pmcd: Version 5.0.2-1, 12 agents, 4 clients
  pmda: root pmcd proc pmproxy xfs linux nfsclient mmv kvm mssql
        jbd2 dm
pmlogger: primary logger: /var/log/pcp/pmlogger/rhel82-2.local/20200326.16.31
pmie: primary engine: /var/log/pcp/pmie/rhel82-2.local/pmie.log
```

추가 리소스

- **Microsoft SQL Server용 Performance Co-hiera** 사용에 대한 자세한 내용은 이 [Red Hat developers](#) 블로그 게시물을 참조하십시오.

20장. MICROSOFT SQL SERVER ANSIBLE 역할을 사용하여 MICROSOFT SQL SERVER 구성

관리자는 **Microsoft SQL Server Ansible** 역할을 사용하여 **Microsoft SQL Server(SQL Server)**를 설치, 구성 및 시작할 수 있습니다. **Microsoft SQL Server Ansible** 역할은 운영 체제를 최적화하여 **SQL Server**의 성능 및 처리량을 향상시킵니다. 이 역할은 **SQL Server** 워크로드를 실행하는 권장 설정으로 **RHEL** 호스트의 구성을 단순화하고 자동화합니다.

20.1. 사전 요구 사항

- **2GB RAM**
- **SQL ServerSQL Server** 구성하려는 관리 노드에 대한 루트 액세스 **root access to the managed node where you want to configure SQL ServerSQL Server**

- 사전 구성된 방화벽

mssql_tcp_port 변수로 설정된 **SQL ServerSQL Server TCP** 포트에서 연결을 활성화해야 합니다. 이 변수를 정의하지 않으면 역할은 기본적으로 **TCP** 포트 번호 **1443**로 설정됩니다.

새 포트를 추가하려면 다음을 사용합니다.

```
# firewall-cmd --add-port=xxxx/tcp --permanent
# firewall-cmd --reload
```

xxxx를 **TCP** 포트 번호로 교체한 다음 방화벽 규칙을 다시 로드합니다.

- **선택 사항:** **SQL ServerSQL Server**에 입력할 수 있는 **SQL** 문과 프로시저가 포함된 **.sql** 확장자로 파일을 생성합니다.

20.2. MICROSOFT SQL SERVER ANSIBLE 역할 설치

Microsoft SQL Server Ansible 역할은 **ansible-collection-microsoft-sql** 패키지의 일부입니다.

사전 요구 사항

- 루트 액세스

절차

1. **RHEL 8 AppStream** 리포지토리에서 사용할 수 있는 **Ansible Core**를 설치합니다.

```
# dnf install ansible-core
```

2. **Microsoft SQL Server Ansible** 역할을 설치합니다.

```
# dnf install ansible-collection-microsoft-sql
```

20.3. MICROSOFT SQL SERVER ANSIBLE 역할을 사용하여 SQL SERVER 설치 및 구성

Microsoft SQL Server Ansible 역할을 사용하여 **SQL** 서버를 설치하고 구성할 수 있습니다.

사전 요구 사항

- **Ansible** 인벤토리가 생성됩니다.

절차

1. **.yml** 확장자를 사용하여 파일을 생성합니다. 예를 들면 **mssql-server.yml** 입니다.
2. **.yml** 파일에 다음 내용을 추가합니다.

```
---
- hosts: all
  vars:
    mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula: true
    mssql_accept_microsoft_cli_utilities_for_sql_server_eula: true
    mssql_accept_microsoft_sql_server_standard_eula: true
    mssql_password: <password>
    mssql_edition: Developer
    mssql_tcp_port: 1443
  roles:
    - microsoft.sql.server
```

`<password>`를 SQL Server 암호로 바꿉니다.

3.

`mssql-server.yml` ansible 플레이북을 실행합니다.

```
# ansible-playbook mssql-server.yml
```

20.4. TLS 변수

다음 변수는 TLS(Transport Level Security)를 구성하는 데 사용할 수 있습니다.

표 20.1. TLS 역할 변수

역할 변수	설명
<code>mssql_tls_enable</code>	이 변수는 TLS 암호화를 활성화 또는 비활성화합니다. Microsoft SQL Server Ansible 역할은 변수가 true 로 설정된 경우 다음 작업을 수행합니다. - SQL Server의 <code>/etc/pki/tls/certs/</code> 에 TLS 인증서를 복사 - SQL Server에서 개인 키를 <code>/etc/pki/tls/private/</code> 에 복사 - TLS 인증서 및 개인 키를 사용하여 연결을 암호화하도록 SQL Server 구성  참고 Ansible 제어 노드에 TLS 인증서 및 개인 키가 있어야 합니다. false 로 설정하면 TLS 암호화가 비활성화됩니다. 이 역할은 기존 인증서 및 개인 키 파일을 제거하지 않습니다.
<code>mssql_tls_cert</code>	이 변수를 정의하려면 TLS 인증서 파일의 경로를 입력합니다.
<code>mssql_tls_private_key</code>	이 변수를 정의하려면 개인 키 파일의 경로를 입력합니다.
<code>mssql_tls_version</code>	사용할 TSL 버전을 선택하려면 이 변수를 정의합니다. 기본값은 1.2입니다.

역할 변수	설명
mssql_tls_force	호스트의 인증서 및 개인 키 파일을 교체하려면 이 변수를 true 로 설정합니다. 파일은 /etc/pki/tls/certs/ 및 /etc/pki/tls/private/ 디렉터리에 있어야 합니다. 기본값은 false 입니다.

20.5. MLSERVICES에 대한 EULA 수락

필요한 **SQL Server SQL Server Machine Learning Services(MLServices)**를 설치하려면 **Python** 및 **R** 패키지의 오픈 소스 배포판에 대해 모든 **EULA**를 수락해야 합니다.

라이선스 용어에 대해서는 **/usr/share/doc/mssql-server** 를 참조하십시오.

표 20.2. SQL Server Machine Learning Services EULA 변수

역할 변수	설명
mssql_accept_microsoft_sql_server_standard_eula	이 변수는 mssql-conf 패키지 설치에 대한 용어 및 조건을 수락할지 여부를 결정합니다. 약관에 동의하려면 이 변수를 true 로 설정합니다. 기본값은 false 입니다.

20.6. MICROSOFT ODBC 17에 대한 EULA 수락

Microsoft Open Database Connectivity(ODBC) 드라이버를 설치하려면 모든 **EULA**를 수락해야 합니다.

라이선스 용어에 대해서는 **/usr/share/doc/msodbcsql17/LICENSE.txt** 및 **/usr/share/doc/mssql-tools/LICENSE.txt** 를 참조하십시오.

표 20.3. Microsoft ODBC 17 EULA 변수

역할 변수	설명
-------	----

역할 변수	설명
mssql_accept_microsoft_odbc_driver_17_for_sql_server_eula	이 변수는 msodbcsql17 패키지를 설치하기 위한 용어 및 조건을 수락할지 여부를 결정합니다. 약관에 동의하려면 이 변수를 true 로 설정합니다. 기본값은 false입니다.
mssql_accept_microsoft_cli_utilities_for_sql_server_eula	이 변수는 mssql-tools 패키지 설치에 대한 용어 및 조건을 수락할지 여부를 결정합니다. 약관에 동의하려면 이 변수를 true 로 설정합니다. 기본값은 false입니다.

21장. 터미널 세션 기록 RHEL 시스템 역할을 사용하여 세션 기록 시스템 구성

터미널 세션 기록 RHEL 시스템 역할을 기록하면 Red Hat Ansible Automation Platform을 사용하여 RHEL에서 터미널 세션 레코딩 시스템을 구성할 수 있습니다.

21.1. 터미널 세션 시스템 역할 기록

RHEL 시스템 역할을 기록하는 Terminal Session Recording RHEL System Role을 사용하여 RHEL에서 터미널 세션 레코딩을 위해 RHEL 시스템을 구성할 수 있습니다.

SSSD 서비스를 통해 사용자 또는 사용자 그룹별로 기록이 수행되도록 구성할 수 있습니다.

추가 리소스

- RHEL의 세션 기록에 대한 자세한 내용은 [기록 세션](#) 을 참조하십시오.

21.2. 터미널 세션 기록 시스템 역할의 구성 요소 및 매개변수

세션 레코드 솔루션에는 다음과 같은 구성 요소가 있습니다.

- tlog 유틸리티
- SSSD(System Security Services Daemon)
- 선택 사항: 웹 콘솔 인터페이스

터미널 세션 기록 RHEL 시스템 역할에 사용되는 매개변수는 다음과 같습니다.

역할 변수	설명
tlog_use_sssd (기본값: yes)	기록된 사용자 또는 그룹을 관리하는 기본 방법인 SSSD를 사용하여 세션 레코딩 구성
tlog_scope_sssd (default: none)	SSSD 레코딩 범위 설정 - 모든 / 일부 / none

역할 변수	설명
tlog_users_sssd (default: [])	생성할 YAML 사용자 목록
tlog_groups_sssd (default: [])	기록할 그룹의 YAML 목록

- **tlog**에 사용되는 매개 변수 및 터미널 세션 기록 시스템 역할에 대한 자세한 내용은 `/usr/share/ansible/roles/rhel-system-roles.tlog/README.md` 파일을 참조하십시오.

21.3. RHEL 시스템 역할 기록 터미널 세션 배포

다음 단계에 따라 **Ansible** 플레이북을 준비하고 적용하여 세션 기록 데이터를 **systemd** 저널에 기록하도록 **RHEL** 시스템을 구성합니다.

사전 요구 사항

- 제어 노드에서 액세스할 **SSH** 키를 터미널 세션 기록 시스템 역할을 구성할 대상 시스템으로 설정했습니다.
- 터미널 세션 기록 시스템 역할을 구성하려는 시스템이 하나 이상 있습니다.
- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- **rhel-system-roles** 패키지는 제어 시스템에 설치됩니다.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploy session recording
  hosts: all
  vars:
    tlog_scope_sssd: some
    tlog_users_sssd:
      - recorded-user
```

```
roles:
  - rhel-system-roles.tlog
```

여기서,

- **tlog_scope_sssd:**
 - 일부는 일부 또는 **none** 이 아닌 특정 사용자와 그룹만 기록하려는 경우 도 있습니다.
- **tlog_users_sssd:**
 - **recorded-user** 는 세션을 기록할 사용자를 지정합니다. 이 경우 사용자를 추가하지 않습니다. 사용자를 직접 설정해야 합니다.

2. 필요한 경우 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

결과적으로 **Playbook**은 사용자가 지정한 시스템에 **tlog RHEL** 시스템 역할을 설치합니다. 역할에는 사용자의 로그인 셸 역할을 하는 터미널 세션 I/O 로깅 프로그램인 **tlog-rec-session** 이 포함됩니다. 또한 사용자가 정의한 사용자 및 그룹에서 사용할 수 있는 **SSSD** 구성 드롭 파일을 생성합니다. **SSSD**는 이러한 사용자 및 그룹을 구문 분석하고 읽고 사용자 셸을 **tlog-rec-session** 으로 교체합니다. 또한 **cockpit** 패키지가 시스템에 설치된 경우 **Playbook**은 웹 콘솔 인터페이스에서 레코딩을 보고 재생할 수 있는 **Cockpit** 모듈인 **cockpit-session-recording** 패키지도 설치합니다.

검증 단계

SSSD 구성 드롭 파일이 시스템에서 생성되었는지 확인하려면 다음 단계를 수행하십시오.

1. **SSSD** 구성 드롭 파일이 생성된 폴더로 이동합니다.

```
# cd /etc/sss/conf.d
```

2.

파일 내용을 확인합니다.

```
# cat /etc/sss/conf.d/sss-session-recording.conf
```

파일에 플레이북에서 설정한 매개변수가 포함되어 있음을 확인할 수 있습니다.

21.4. 그룹 또는 사용자 목록을 제외하고 RHEL 시스템 역할을 기록하는 터미널 세션 기록

터미널 세션 기록 시스템 역할을 사용하여 **exclude_users** 및 **exclude_groups** 를 **SSSD** 세션 레코딩 구성 옵션을 지원할 수 있습니다. 다음 단계에 따라 사용자 또는 그룹을 기록하고 **systemd** 저널에 로그인 하지 못하도록 **RHEL** 시스템을 구성하고 적용하도록 **Ansible** 플레이북을 준비하고 적용합니다.

사전 요구 사항

- 터미널 세션 기록 시스템 역할을 구성하려는 제어 노드에서 대상 시스템으로 **SSH** 키를 설정했습니다.
- 터미널 세션 기록 시스템 역할을 구성하려는 시스템이 하나 이상 있습니다.
- **Ansible Core** 패키지는 제어 시스템에 설치됩니다.
- **rhel-system-roles** 패키지는 제어 시스템에 설치됩니다.

절차

1.

다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- name: Deploy session recording excluding users and groups
  hosts: all
  vars:
    tlog_scope_sss: all
    tlog_exclude_users_sss:
      - jeff
      - james
    tlog_exclude_groups_sss:
```

```
- admins
```

```
roles:
```

```
- rhel-system-roles.tlog
```

여기서,

-

tlog_scope_sssd:

-

all : 모든 사용자 및 그룹을 기록하도록 지정합니다.

-

tlog_exclude_users_sssd:

-

사용자 이름: 세션 기록에서 제외하려는 사용자의 사용자 이름을 지정합니다.

-

tlog_exclude_groups_sssd:

-

admins 는 세션 레코딩에서 제외할 그룹을 지정합니다.

2.

필요한 경우 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i IP_Address /path/to/file/playbook.yml -v
```

결과적으로 **Playbook**은 사용자가 지정한 시스템에 **tlog RHEL** 시스템 역할을 설치합니다. 역할에는 사용자의 로그인 셸 역할을 하는 터미널 세션 I/O 로깅 프로그램인 **tlog-rec-session** 이 포함됩니다. 또한 제외된 것으로 정의된 것을 제외하고 사용자와 그룹에서 사용할 수 있는 **/etc/sss/conf.d/sss-session-recordedding.conf** SSSD 구성 드롭 파일을 생성합니다. SSSD는 이러한 사용자 및 그룹을 구문 분석하고 읽고 사용자 셸을 **tlog-rec-session** 으로 교체합니다. 또한 **cockpit** 패키지가 시스템에 설치된 경우 플레이북은 웹 콘솔 인터페이스에서 레코딩을 보고 재생할 수 있는 **Cockpit** 모듈인 **cockpit-session-Recording** 패키지도 설치합니다.

검증 단계

SSSD 구성 드롭 파일이 시스템에서 생성되었는지 확인하려면 다음 단계를 수행하십시오.

1. **SSSD** 구성 드롭 파일이 생성된 폴더로 이동합니다.

```
# cd /etc/sss/conf.d
```

2. 파일 내용을 확인합니다.

```
# cat sssd-session-recording.conf
```

파일에 플레이북에서 설정한 매개변수가 포함되어 있음을 확인할 수 있습니다.

추가 리소스

- [/usr/share/doc/rhel-system-roles/tlog/](#) 및 [/usr/share/ansible/roles/rhel-system-roles.tlog/](#) 디렉토리를 참조하십시오.
- [CLI에서 배포된 터미널 세션 기록 시스템 역할을 사용하여 세션을 기록합니다.](#)

21.5. CLI에서 배포된 터미널 세션 기록 시스템 역할을 사용하여 세션 기록

지정한 시스템에 **Terminal Session Recording System Role**을 배포한 후 **CLI**(명령줄 인터페이스)를 사용하여 사용자 터미널 세션을 기록할 수 있습니다.

사전 요구 사항

- 대상 시스템에 터미널 세션 기록 시스템 역할을 배포했습니다.
- **SSSD** 구성 드롭 파일이 [/etc/sss/conf.d](#) 디렉토리에 생성되었습니다. [터미널 세션 기록 RHEL 시스템 역할 배포](#)를 참조하십시오.

절차

1. 사용자를 생성하고 이 사용자의 암호를 할당합니다.

```
# useradd recorded-user
# passwd recorded-user
```

2. 방금 만든 사용자로 시스템에 로그인합니다.

```
# ssh recorded-user@localhost
```

3. 시스템이 **yes** 또는 **no**를 입력하여 인증할 것인지 묻는 메시지가 표시되면 **"yes"**를 입력합니다.

4. **recorded-user**의 암호를 삽입합니다.

시스템은 기록되고 있는 세션에 대한 메시지를 표시합니다.

```
ATTENTION! Your session is being recorded!
```

5. 세션 레코딩을 완료한 후 다음을 입력합니다.

```
# exit
```

시스템은 사용자가 로그아웃한 후 **localhost**와의 연결을 종료합니다.

결과적으로 사용자 세션이 기록되고 저장되고 저널을 사용하여 재생할 수 있습니다.

검증 단계

저널에서 기록된 세션을 보려면 다음 단계를 수행합니다.

1. 아래 명령을 실행하십시오.

```
# journalctl -o verbose -r
```

2. **tlog-rec** 기록된 저널 항목의 **MESSAGE** 필드를 검색합니다.

```
# journalctl -xel _EXE=/usr/bin/tlog-rec-session
```

21.6. CLI를 사용하여 기록된 세션 모니터링

CLI(명령줄 인터페이스)를 사용하여 저널에서 사용자 세션 레코딩을 재생할 수 있습니다.

사전 요구 사항

- 사용자 세션을 기록했습니다. **CLI에서 배포된 터미널 세션 기록 시스템 역할을 사용하여** 세션 기록을 참조하십시오.

절차

1. **CLI** 터미널에서 사용자 세션 레코딩을 재생합니다.

```
# journalctl -o verbose -r
```

2. **tlog** 레코딩을 검색합니다.

```
$ /tlog-rec
```

다음과 같은 세부 정보를 볼 수 있습니다.

- 사용자 세션 기록의 사용자 이름입니다.
- **out_txt** 필드, 기록된 세션의 원시 출력 인코딩
- 식별자 번호 **TLOG_REC=ID_number**

3. 식별자 번호 **TLOG_REC=ID_number**를 복사합니다.
4. **TLOG_REC=ID_number**를 사용하여 레코딩을 재생합니다.

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

결과적으로 재생되는 사용자 세션 기록 터미널 출력을 볼 수 있습니다.

22장. 시스템 역할을 사용하여 고가용성 클러스터 구성

HA 클러스터 시스템 역할을 사용하면 **Pacemaker** 고가용성 클러스터 리소스 관리자를 사용하는 고가용성 클러스터를 구성하고 관리할 수 있습니다.



참고

HA 시스템 역할은 현재 **SBD**를 지원하지 않습니다.

22.1. HA 클러스터 시스템 역할 변수

HA 클러스터 시스템 역할 플레이북에서는 클러스터 배포 요구 사항에 따라 고가용성 클러스터의 변수를 정의합니다.

HA 클러스터 시스템 역할에 설정할 수 있는 변수는 다음과 같습니다.

`ha_cluster_enable_repos`

HA Cluster System 역할에 필요한 패키지가 포함된 리포지토리를 활성화하는 부울 플래그입니다. 이 변수의 기본값인 **yes**로 설정하면 클러스터 구성원 또는 시스템 역할이 실패할 때 사용할 시스템에서 **RHEL** 및 **RHEL** 고가용성 애드온에 대한 활성화 서브스크립션 적용 범위가 있어야 합니다.

`ha_cluster_cluster_present`

부울 플래그 **yes**로 설정하면 역할에 전달된 변수에 따라 **HA** 클러스터가 호스트에 구성되는지 확인합니다. 역할에 지정되지 않고 역할에서 지원되지 않는 클러스터 구성이 손실됩니다.

`ha_cluster_cluster_present`이 **no**로 설정되어 있으면 모든 **HA** 클러스터 구성이 대상 호스트에서 제거됩니다.

이 변수의 기본값은 **yes**입니다.

다음 예제 플레이북은 **node1** 및 **node2**의 모든 클러스터 구성을 제거합니다.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_present: no
```

```
roles:
- rhel-system-roles.ha_cluster
```

ha_cluster_start_on_boot

부팅 시 시작되도록 클러스터 서비스를 구성할지 여부를 결정하는 부울 플래그입니다. 이 변수의 기본값은 **yes** 입니다.

ha_cluster_fence_agent_packages

설치할 펜스 에이전트 패키지 목록입니다. 이 변수의 기본값은 **fence-agents-all,fence-virt** 입니다.

ha_cluster_extra_packages

설치할 추가 패키지 목록입니다. 이 변수의 기본값은 패키지가 아닙니다.

이 변수는 역할(예: 사용자 정의 리소스 에이전트)에서 자동으로 설치되지 않은 추가 패키지를 설치하는 데 사용할 수 있습니다.

이 목록의 멤버로 펜스 에이전트를 지정할 수 있습니다. 그러나 기본값을 재정의하도록 **ha_cluster_fence_agent_packages** 는 펜스 에이전트를 지정하는 데 사용할 권장 역할 변수입니다.

ha_cluster_hacluster_password

hacluster 사용자의 암호를 지정하는 문자열 값입니다. **hacluster** 사용자는 클러스터에 대한 전체 액세스 권한을 갖습니다. **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 **vault**에서 암호를 암호화하는 것이 좋습니다. 기본 암호 값이 없으며 이 변수를 지정해야 합니다.

ha_cluster_corosync_key_src

Corosync 통신을 위한 인증 및 암호화 키인 **Corosync authkey** 파일의 경로입니다. 각 클러스터에 고유한 **authkey** 값이 있는 것이 좋습니다. 키는 256바이트의 임의 데이터여야 합니다.

이 변수의 키를 지정하는 경우 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 **vault** 암호화 키를 암호화하는 것이 좋습니다.

키가 지정되지 않은 경우 노드에 이미 존재하는 키가 사용됩니다. 노드에 동일한 키가 없는 경우 모든 노드에 동일한 키가 있도록 한 노드의 키가 다른 노드에 배포됩니다. 노드가 키가 없는 경우 새 키가 생성되어 노드에 배포됩니다.

이 변수가 설정되면 이 키에 대해 **ha_cluster_regenerate_keys** 가 무시됩니다.

이 변수의 기본값은 **null**입니다.

ha_cluster_pacemaker_key_src

Pacemaker 통신의 인증 및 암호화 키인 **Pacemaker authkey** 파일의 경로입니다. 각 클러스터에 고유한 **authkey** 값이 있는 것이 좋습니다. 키는 256바이트의 임의 데이터여야 합니다.

이 변수의 키를 지정하는 경우 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 **vault** 암호화 키를 암호화하는 것이 좋습니다.

키가 지정되지 않은 경우 노드에 이미 존재하는 키가 사용됩니다. 노드에 동일한 키가 없는 경우 모든 노드에 동일한 키가 있도록 한 노드의 키가 다른 노드에 배포됩니다. 노드가 키가 없는 경우 새 키가 생성되어 노드에 배포됩니다.

이 변수가 설정되면 이 키에 대해 **ha_cluster_regenerate_keys**가 무시됩니다.

이 변수의 기본값은 **null**입니다.

ha_cluster_fence_virt_key_src

fence-virt 또는 **fence-xvm** 펜스 에이전트의 인증 키 위치인 **fence-virt** 또는 **fence-xvm** 사전 공유 키 파일의 경로입니다.

이 변수의 키를 지정하는 경우 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 **vault** 암호화 키를 암호화하는 것이 좋습니다.

키가 지정되지 않은 경우 노드에 이미 존재하는 키가 사용됩니다. 노드에 동일한 키가 없는 경우 모든 노드에 동일한 키가 있도록 한 노드의 키가 다른 노드에 배포됩니다. 노드가 키가 없는 경우 새 키가 생성되어 노드에 배포됩니다. **HA Cluster System Role**에서 이러한 방식으로 새 키를 생성하는 경우 해당 키를 노드 하이퍼바이저에 복사하여 펜싱이 작동하는지 확인해야 합니다.

이 변수가 설정되면 이 키에 대해 **ha_cluster_regenerate_keys**가 무시됩니다.

이 변수의 기본값은 **null**입니다.

ha_cluster_pcsd_public_key_src, ha_cluster_pcsd_private_key_src

pcsd TLS 인증서 및 개인 키의 경로입니다. 이를 지정하지 않으면 노드에 이미 존재하는 인증서-키 쌍이 사용됩니다. 인증서-키 쌍이 없으면 임의의 새 키가 생성됩니다.

이 변수에 대해 개인 키 값을 지정하는 경우 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 **vault가 키를 암호화하는** 것이 좋습니다.

이러한 변수가 설정되면 이 인증서-키 쌍에서 **ha_cluster_regenerate_keys**가 무시됩니다.

이러한 변수의 기본값은 **null**입니다.

ha_cluster_regenerate_keys

부울 플래그 **yes**로 설정하면 사전 공유 키와 **TLS** 인증서가 다시 생성됩니다. 키와 인증서를 다시 생성할 시기에 대한 자세한 내용은 **ha_cluster_corosync_key_src, ha_cluster_pacemaker_key_src, ha_cluster_fence_virt_key_src, ha_cluster_pcsd_key_src**에 대한 설명을 참조하십시오.

이 변수의 기본값은 **no**입니다.

ha_cluster_pcs_permission_list

pcsd를 사용하여 클러스터를 관리할 수 있는 권한을 구성합니다. 이 변수로 구성하는 항목은 다음과 같습니다.

- 사용자 또는 그룹 유형
- 이름 - 사용자 또는 그룹 이름
- **allow_list** - 지정된 사용자 또는 그룹에 대해 허용된 작업:
 - 읽기 - 클러스터 상태 및 설정 보기
 - 쓰기 - 권한 및 **ACL**을 제외한 클러스터 설정 수정

- 부여 - 클러스터 권한 및 **ACL** 수정
- 전체 - 노드 추가 및 제거 및 키와 인증서에 대한 액세스 권한을 포함하여 클러스터에 대한 무제한 액세스 권한

:: **ha_cluster_pcs_permission_list** 변수와 해당 기본값은 다음과 같습니다.

```
ha_cluster_pcs_permission_list:
- type: group
  name: hacluster
  allow_list:
    - grant
    - read
    - write
```

ha_cluster_cluster_name

클러스터의 이름입니다. 이는 기본값 **my-cluster** 가 있는 문자열 값입니다.

ha_cluster_cluster_properties

Pacemaker 클러스터 전체 구성의 클러스터 속성 세트 목록입니다. 하나의 클러스터 속성 세트만 지원됩니다.

클러스터 속성 세트의 구조는 다음과 같습니다.

```
ha_cluster_cluster_properties:
- attrs:
    - name: property1_name
      value: property1_value
    - name: property2_name
      value: property2_value
```

기본적으로 속성은 설정되어 있지 않습니다.**By default, no properties are set.**

다음 예제 플레이북은 **node1** 및 **node2** 로 구성된 클러스터를 구성하고 **stonith-enabled** 및 **no-quorum-policy** 클러스터 속성을 설정합니다.

```
- hosts: node1 node2
  vars:
    ha_cluster_cluster_name: my-new-cluster
    ha_cluster_hacluster_password: password
    ha_cluster_cluster_properties:
```

```
- attrs:
  - name: stonith-enabled
    value: 'true'
  - name: no-quorum-policy
    value: stop

roles:
  - rhel-system-roles.ha_cluster
```

ha_cluster_resource_primitives

이 변수는 **stonith** 리소스를 포함하여 시스템 역할에서 구성된 **pacemaker** 리소스를 정의합니다. 각 리소스에 대해 구성할 수 있는 항목은 다음과 같습니다.

- ID (필수) - 리소스의 ID입니다.
- agent (mandatory) - 리소스 또는 **stonith** 에이전트의 이름(예: **ocf:pacemaker:Dummy** 또는 **stonith:fence_xvm**)입니다. **stonith: stonith** 에이전트를 지정해야 합니다. 리소스 에이전트의 경우 **ocf:pacemaker:Dummy** 대신 **Dummy** 와 같은 짧은 이름을 사용할 수 있습니다. 그러나 짧은 이름이 같은 여러 에이전트가 설치되어 있으면 사용할 에이전트를 결정할 수 없으므로 역할이 실패합니다. 따라서 리소스 에이전트를 지정할 때 전체 이름을 사용하는 것이 좋습니다.
- instance_attrs (선택 사항) - 리소스의 인스턴스 속성 세트 목록입니다. 현재는 하나의 세트만 지원됩니다. 특성의 정확한 이름과 값은 필수인지 여부, 리소스 또는 **stonith** 에이전트에 따라 달라집니다.
- meta_attrs (선택 사항) - 리소스의 메타 속성 세트 목록입니다. 현재는 하나의 세트만 지원됩니다.
- operations (선택 사항) - 리소스의 작업 목록.
 - 작업 (필수) - **pacemaker** 및 리소스 또는 **stonith** 에이전트에서 정의한 작업 작업입니다.
 - attrs (필수) - 운영 옵션, 하나 이상의 옵션을 지정해야 합니다.

:: HA 클러스터 시스템 역할을 사용하여 구성하는 리소스 정의의 구조는 다음과 같습니다.

```

- id: resource-id
  agent: resource-agent
  instance_attrs:
    - attrs:
        - name: attribute1_name
          value: attribute1_value
        - name: attribute2_name
          value: attribute2_value
  meta_attrs:
    - attrs:
        - name: meta_attribute1_name
          value: meta_attribute1_value
        - name: meta_attribute2_name
          value: meta_attribute2_value
  operations:
    - action: operation1-action
      attrs:
        - name: operation1_attribute1_name
          value: operation1_attribute1_value
        - name: operation1_attribute2_name
          value: operation1_attribute2_value
    - action: operation2-action
      attrs:
        - name: operation2_attribute1_name
          value: operation2_attribute1_value
        - name: operation2_attribute2_name
          value: operation2_attribute2_value

```

기본적으로 리소스는 정의되지 않습니다.

리소스 구성을 포함하는 **HA Cluster System Role Playbook**의 예는 [펜싱 및 리소스를 사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_resource_groups

이 변수는 시스템 역할을 통해 구성된 **pacemaker** 리소스 그룹을 정의합니다. 각 리소스 그룹에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **ID (필수)** - 그룹의 ID입니다.
- **resources (필수)** - 그룹의 리소스 목록입니다. 각 리소스는 **ID**에서 참조하며 **ha_cluster_resource_primitives** 변수에 리소스를 정의해야 합니다. 하나 이상의 리소스가 나열되어야 합니다.
- **meta_attrs (선택 사항)** - 그룹의 메타 속성 세트 목록입니다. 현재는 하나의 세트만 지원

됩니다.

:: HA 클러스터 시스템 역할을 사용하여 구성하는 리소스 그룹 정의의 구조는 다음과 같습니다.

```
ha_cluster_resource_groups:
  - id: group-id
    resource_ids:
      - resource1-id
      - resource2-id
    meta_attrs:
      - attrs:
          - name: group_meta_attribute1_name
            value: group_meta_attribute1_value
          - name: group_meta_attribute2_name
            value: group_meta_attribute2_value
```

기본적으로 리소스 그룹이 정의되지 않습니다.

리소스 그룹 구성을 포함하는 HA Cluster System Role Playbook의 예는 [펜싱 및 리소스를 사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_resource_clones

이 변수는 시스템 역할을 통해 구성된 **pacemaker** 리소스 복제본을 정의합니다. 리소스 복제에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_id** (필수) - 복제할 리소스입니다. 리소스는 **ha_cluster_resource_primitives** 변수 또는 **ha_cluster_resource_groups** 변수에 정의해야 합니다.
- **promotable** (선택 사항) - 생성할 리소스 복제가 프로모션 가능한 복제인지(예: **yes** 또는 **no**) 임을 나타냅니다.
- **ID** (선택 사항) - 복제본의 사용자 정의 ID입니다. ID를 지정하지 않으면 생성됩니다. 클러스터에서 이 옵션을 지원하지 않으면 경고가 표시됩니다.
- **meta_attrs** (선택 사항) - 클론의 메타 속성 세트 목록. 현재는 하나의 세트만 지원됩니다.

:: HA 클러스터 시스템 역할을 사용하여 구성하는 리소스 복제 정의의 구조는 다음과 같

습니다.

```
ha_cluster_resource_clones:
  - resource_id: resource-to-be-cloned
    promotable: yes
    id: custom-clone-id
    meta_attrs:
      - attrs:
          - name: clone_meta_attribute1_name
            value: clone_meta_attribute1_value
          - name: clone_meta_attribute2_name
            value: clone_meta_attribute2_value
```

기본적으로 리소스 복제가 정의되지 않습니다.

리소스 복제 구성을 포함하는 **HA Cluster System Role Playbook**의 예는 [펜싱 및 리소스를 사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_constraints_location

이 변수는 리소스 위치 제약 조건을 정의합니다. 리소스 위치 제한 조건은 리소스를 실행할 수 있는 노드를 나타냅니다. 리소스 ID 또는 둘 이상의 리소스와 일치시킬 수 있는 패턴으로 지정된 리소스를 지정할 수 있습니다. 노드 이름 또는 규칙으로 노드를 지정할 수 있습니다.

리소스 위치 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- 리소스 (필수) - 제약 조건이 적용되는 리소스의 사양입니다.
- 노드 (필수) - 리소스에서 선호하거나 피해야 하는 노드의 이름입니다.
- ID (선택 사항) - 제약 조건의 ID입니다. 지정하지 않으면 자동으로 생성됩니다.
- 옵션 (선택 사항) - 이름-값 사전 목록입니다.
 - **core** - 제약 조건의 가중치를 설정합니다.
 - 양수 점수 값은 리소스에서 노드에서 실행되는 것을 선호하는 것을 의미합니다.

- 음수 점수 값은 리소스가 노드에서 실행되지 않도록 합니다.
 - 점수 값 **-INFINITY** 는 리소스가 노드에서 실행되지 않도록 해야 함을 의미합니다.
 - 점수가 지정되지 않은 경우 점수 값은 기본값인 **INFINITY** 입니다.
- :: 기본적으로 리소스 위치 제약 조건이 정의되지 않습니다.

리소스 ID와 노드 이름을 지정하는 리소스 위치 제한 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_location:
- resource:
  id: resource-id
  node: node-name
  id: constraint-id
  options:
  - name: score
    value: score-value
  - name: option-name
    value: option-value
```

리소스 위치 제약 조건에 대해 구성하는 항목은 리소스 사양 자체를 제외하고 리소스 ID를 지정하는 리소스 위치 제약 조건에 대해 구성하는 항목과 같습니다. 리소스 사양에 지정하는 항목은 다음과 같습니다.

- 패턴 (필수) - **POSIX** 확장 정규식 리소스 ID가 와 일치합니다.

:: 리소스 패턴 및 노드 이름을 지정하는 리소스 위치 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_location:
- resource:
  pattern: resource-pattern
  node: node-name
  id: constraint-id
  options:
  - name: score
```

```

value: score-value
- name: resource-discovery
  value: resource-discovery-value

```

리소스 ID와 규칙을 지정하는 리소스 위치 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- 리소스 (필수) - 제약 조건이 적용되는 리소스의 사양입니다.
 - ID (필수) - 리소스 ID.
 - role (선택 사항) - 제약 조건이 제한된 리소스 역할입니다. 시작됨, 프로모션되지 않음.
- 규칙 (필수) - pcs 구문을 사용하여 작성된 제약 조건 규칙. 자세한 내용은 pcs(8) 매뉴얼 페이지의 제약 조건 위치 섹션을 참조하십시오.
- 지정할 다른 항목은 규칙을 지정하지 않는 리소스 제약 조건과 동일한 의미를 갖습니다.

:: 리소스 ID와 규칙을 지정하는 리소스 위치 제약 조건의 구조는 다음과 같습니다.

```

ha_cluster_constraints_location:
- resource:
  id: resource-id
  role: resource-role
  rule: rule-string
  id: constraint-id
  options:
  - name: score
    value: score-value
  - name: resource-discovery
    value: resource-discovery-value

```

리소스 위치 제약 조건을 지정하는 리소스 위치 제약 조건에 대해 구성하는 항목은 리소스 ID와 리소스 사양 자체를 제외하고 규칙을 지정하는 리소스 위치 제약 조건에 대해 구성하는 항목과 동일합니다. 리소스 사양에 지정하는 항목은 다음과 같습니다.

- 패턴 (필수) - POSIX 확장 정규식 리소스 ID가 와 일치합니다.

:: 리소스 패턴을 지정하는 리소스 위치 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_location:
- resource:
  pattern: resource-pattern
  role: resource-role
  rule: rule-string
  id: constraint-id
  options:
  - name: score
    value: score-value
  - name: resource-discovery
    value: resource-discovery-value
```

리소스 제약 조건으로 클러스터를 생성하는 **ha_cluster** 시스템 역할 플레이북의 예는 리소스 제약 조건을 [사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_constraints_colocation

이 변수는 리소스 공동 배치 제약 조건을 정의합니다. 리소스 공동 배치 제한 조건은 한 리소스의 위치가 다른 리소스의 위치에 따라 달라지는 것을 나타냅니다. 공동 배치 제한에는 두 리소스에 대한 간단한 공동 배치 제한 조건과 여러 리소스에 대한 설정된 배치 제약 조건의 두 가지 유형이 있습니다.

간단한 리소스 공동 배치 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_follower (mandatory) - resource_leader** 를 기준으로 배치되어야 하는 리소스입니다.
 - **ID (필수) - 리소스 ID.**
 - **role (선택 사항) - 제약 조건이 제한된 리소스 역할입니다. 시작됨,프로모션되지 않음.**
- **resource_leader (mandatory) - 클러스터는 이 리소스를 먼저 배치할 위치를 결정하고 resource_follower 를 배치할 위치를 결정합니다.**
 - **ID (필수) - 리소스 ID.**
 - **role (선택 사항) - 제약 조건이 제한된 리소스 역할입니다. 시작됨,프로모션되지 않음.**

음.

- ID (선택 사항) - 제약 조건의 ID입니다. 지정하지 않으면 자동으로 생성됩니다.
- 옵션 (선택 사항) - 이름-값 사전 목록입니다.
 - **core** - 제약 조건의 가중치를 설정합니다.
 - 양수 점수 값은 동일한 노드에서 리소스를 실행해야 함을 나타냅니다.
 - 음수 점수 값은 다른 노드에서 리소스를 실행해야 함을 나타냅니다.
 - 점수 값은 **+INFINITY** 가 동일한 노드에서 실행되어야 하는 리소스를 나타냅니다.
 - 점수 값 **-INFINITY** 는 다른 노드에서 리소스를 실행해야 함을 나타냅니다.
 - 점수가 지정되지 않은 경우 점수 값은 기본값은 **INFINITY** 입니다.

:: 기본적으로 리소스 공동 배치 제약 조건이 정의되지 않습니다.**By default no resource colocation constraints are defined.**

간단한 리소스 공동 배치 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_colocation:
- resource_follower:
  id: resource-id1
  role: resource-role1
resource_leader:
  id: resource-id2
  role: resource-role2
id: constraint-id
options:
- name: score
  value: score-value
- name: option-name
  value: option-value
```

-

리소스 세트 공동 배치 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_sets** (필수) - 리소스 세트 목록입니다.
 - **resource_ids** (필수) - 세트의 리소스 목록입니다.
 - 옵션 (선택 사항) - 이름-값 사전 목록의 제한 조건을 통해 세트의 리소스를 처리하는 방법을 미세 조정합니다.
- **ID** (선택 사항) - 간단한 공동 배치 제약 조건의 것과 동일한 값입니다.
- 옵션 (선택 사항) - 간단한 공동 배치 제약 조건의 경우와 동일한 값입니다.

:: 리소스 세트 배치 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_colocation:
```

```
- resource_sets:
  - resource_ids:
    - resource-id1
    - resource-id2
  options:
    - name: option-name
      value: option-value
id: constraint-id
options:
  - name: score
    value: score-value
  - name: option-name
    value: option-value
```

리소스 제약 조건으로 클러스터를 생성하는 **ha_cluster** 시스템 역할 플레이북의 예는 리소스 제약 조건을 [사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_constraints_order

이 변수는 리소스 순서 제약 조건을 정의합니다. 리소스 순서 제약 조건은 특정 리소스 작업을 수행해야 하는 순서를 나타냅니다. 리소스 순서 제한에는 두 가지 리소스에 대한 간단한 순서 제약 조건, 여러 리소스에 대한 일련의 순서 제약 조건의 두 가지 유형이 있습니다.

간단한 리소스 순서 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_first (필수) - resource_then** 리소스가 의존하는 리소스입니다.
 - **ID (필수) - 리소스 ID.**
 - **action (선택 사항) - resource_then** 리소스에 대해 작업을 시작하기 전에 완료해야 하는 작업입니다. 허용되는 값: **start,stop,promote,demote**.
- **resource_then (필수) - 종속 리소스.**
 - **ID (필수) - 리소스 ID.**
 - **action (선택 사항) - resource_first** 리소스의 작업이 완료된 후에만 리소스를 실행할 수 있는 작업입니다. 허용되는 값: **start,stop,promote,demote**.
- **ID (선택 사항) - 제약 조건의 ID**입니다. 지정하지 않으면 자동으로 생성됩니다.
- **옵션 (선택 사항) - 이름-값** 사전 목록입니다.

:: 기본적으로 리소스 순서 제약 조건이 정의되지 않습니다.

간단한 리소스 순서 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_order:
- resource_first:
  id: resource-id1
  action: resource-action1
  resource_then:
    id: resource-id2
    action: resource-action2
  id: constraint-id
  options:
    - name: score
```

```

value: score-value
- name: option-name
  value: option-value

```

리소스 세트 순서 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_sets** (필수) - 리소스 세트 목록입니다.
 - **resource_ids** (필수) - 세트의 리소스 목록입니다.
 - 옵션 (선택 사항) - 이름-값 사전 목록의 제한 조건을 통해 세트의 리소스를 처리하는 방법을 미세 조정합니다.
- **ID** (선택 사항) - 간단한 순서 제약 조건의 경우와 동일한 값입니다.
- 옵션 (선택 사항) - 간단한 순서 제약 조건에서와 동일한 값입니다.

:: 리소스 세트 순서 제약 조건의 구조는 다음과 같습니다.

```

ha_cluster_constraints_order:
- resource_sets:
  - resource_ids:
    - resource-id1
    - resource-id2
  options:
    - name: option-name
      value: option-value
  id: constraint-id
  options:
    - name: score
      value: score-value
    - name: option-name
      value: option-value

```

리소스 제약 조건으로 클러스터를 생성하는 **ha_cluster** 시스템 역할 플레이북의 예는 리소스 제약 조건을 [사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

ha_cluster_constraints_ticket

이 변수는 리소스 티켓 제약 조건을 정의합니다. 리소스 티켓 제약 조건은 특정 티켓에 의존하는

리소스를 나타냅니다. 리소스 티켓에는 하나의 리소스에 대한 간단한 티켓 제약 조건, 여러 리소스에 대한 티켓 순서 제약 조건의 두 가지 유형이 있습니다.

간단한 리소스 티켓 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- 리소스 (필수) - 제약 조건이 적용되는 리소스의 사양입니다.
 - ID (필수) - 리소스 ID.
 - role (선택 사항) - 제약 조건이 제한된 리소스 역할입니다. 시작됨,프로모션되지 않음.
 - 티켓 (권장) - 리소스가 종속된 티켓의 이름입니다.
 - ID (선택 사항) - 제약 조건의 ID입니다. 지정하지 않으면 자동으로 생성됩니다.
 - 옵션 (선택 사항) - 이름-값 사전 목록입니다.
 - loss-policy (선택 사항) - 티켓이 취소된 경우 리소스에서 수행할 작업입니다.
- :: 기본적으로 리소스 티켓 제약 조건이 정의되지 않습니다.

간단한 리소스 티켓 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_ticket:
- resource:
  id: resource-id
  role: resource-role
ticket: ticket-name
id: constraint-id
options:
- name: loss-policy
  value: loss-policy-value
- name: option-name
  value: option-value
```

리소스 세트 티켓 제약 조건에 대해 구성할 수 있는 항목은 다음과 같습니다.

- **resource_sets (필수)** - 리소스 세트 목록입니다.
 - **resource_ids (필수)** - 세트의 리소스 목록입니다.
 - 옵션 (선택 사항) - 이름-값 사전 목록의 제한 조건을 통해 세트의 리소스를 처리하는 방법을 미세 조정합니다.
- **티켓 (권장)** - 간단한 티켓 제약 조건의 경우와 동일한 값입니다.
- **ID (선택 사항)** - 간단한 티켓 제약 조건에서와 동일한 값입니다.
- **옵션 (선택 사항)** - 간단한 티켓 제약 조건에서와 동일한 값입니다.

:: 리소스 세트 티켓 제약 조건의 구조는 다음과 같습니다.

```
ha_cluster_constraints_ticket:
- resource_sets:
  - resource_ids:
    - resource-id1
    - resource-id2
  options:
    - name: option-name
      value: option-value
ticket: ticket-name
id: constraint-id
options:
  - name: option-name
    value: option-value
```

리소스 제약 조건으로 클러스터를 생성하는 **ha_cluster** 시스템 역할 플레이북의 예는 리소스 제약 조건을 [사용하여 고가용성 클러스터](#) 구성을 참조하십시오.

22.2. HA 클러스터 시스템 역할에 대한 인벤토리 지정

HA 클러스터 시스템 역할 플레이북을 사용하여 **HA** 클러스터를 구성할 때 인벤토리에서 클러스터에 대

한 노드의 이름과 주소를 구성합니다.

인벤토리의 각 노드에 대해 선택적으로 다음 항목을 지정할 수 있습니다.

- **NODE_NAME** - 클러스터에 있는 노드 이름입니다.
- **pcs_address** - pcs 가 노드와 통신하는 데 사용하는 주소입니다. 이름, FQDN 또는 IP 주소일 수 있으며 포트 번호를 포함할 수 있습니다.
- **corosync_addresses** - Corosync에서 사용하는 주소 목록입니다. 특정 클러스터를 구성하는 모든 노드는 동일한 수의 주소와 주소 순서가 있어야 합니다.

다음 예제에서는 **target node1** 및 **node2** . **node2** 가 있는 인벤토리를 보여줍니다. **node1** 및 **node2** 는 정규화된 도메인 이름이어야 하며, 그렇지 않으면 노드의 이름을 **/etc/hosts** 파일을 통해 확인할 수 있어야 합니다.

```
all:
  hosts:
    node1:
      ha_cluster:
        node_name: node-A
        pcs_address: node1-address
        corosync_addresses:
          - 192.168.1.11
          - 192.168.2.11
    node2:
      ha_cluster:
        node_name: node-B
        pcs_address: node2-address:2224
        corosync_addresses:
          - 192.168.1.12
          - 192.168.2.12
```

22.3. 리소스 없이 실행되는 고가용성 클러스터 구성

다음 절차에서는 **HA** 클러스터 시스템 역할을 사용하여 펜싱이 구성되지 않고 리소스가 없는 고가용성 클러스터를 생성합니다.

사전 요구 사항

- 플레이북을 실행할 노드에 **ansible-core** 가 설치되어 있어야 합니다.



참고

ansible-core 를 클러스터 멤버 노드에 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#) 를 참조하십시오.

- 클러스터 구성원으로 사용할 **RHEL**을 실행하는 시스템에는 **RHEL** 및 **RHEL** 고가용성 애드온에 대해 유효한 서브스크립션 적용 범위가 있어야 합니다.



주의

HA Cluster System Role은 지정된 노드의 기존 클러스터 구성을 대체합니다. 역할에 지정되지 않은 설정은 모두 손실됩니다.

절차

1. **HA 클러스터 시스템 역할에 대한 인벤토리 지정에 설명된 대로 클러스터에 노드를 지정하는 인벤토리** 파일을 생성합니다.
2. 플레이북 파일(예: **new-cluster.yml**)을 생성합니다.

다음 예제 플레이북 파일은 펜싱이 구성되지 않고 리소스 없이 실행되는 클러스터를 구성합니다. 프로덕션용 플레이북 파일을 생성할 때 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 암호를 암호화하는 것이 좋습니다.

```
- hosts: node1 node2
vars:
  ha_cluster_cluster_name: my-new-cluster
  ha_cluster_hacluster_password: password
```

```
roles:
- rhel-system-roles.ha_cluster
```

3. 파일을 저장합니다.
4. 1단계에서 생성한 인벤토리 파일 *인벤토리*의 경로를 지정하여 플레이북을 실행합니다.

```
# ansible-playbook -i inventory new-cluster.yml
```

22.4. 펜싱 및 리소스를 사용하여 고가용성 클러스터 구성

다음 절차에서는 **HA Cluster System Role**을 사용하여 펜싱 장치, 클러스터 리소스, 리소스 그룹 및 복제된 리소스를 포함하는 고가용성 클러스터를 생성합니다.

사전 요구 사항

- 플레이북을 실행할 노드에 **ansible-core**가 설치되어 있어야 합니다.



참고

ansible-core를 클러스터 멤버 노드에 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#)를 참조하십시오.

- 클러스터 구성원으로 사용할 **RHEL**을 실행하는 시스템에는 **RHEL** 및 **RHEL** 고가용성 애드온에 대해 유효한 서브스크립션 적용 범위가 있어야 합니다.



주의

HA Cluster System Role은 지정된 노드의 기존 클러스터 구성을 대체합니다. 역할에 지정되지 않은 설정은 모두 손실됩니다.

절차

1. **HA 클러스터 시스템 역할에 대한 인벤토리 지정에 설명된 대로 클러스터에 노드를 지정하는 인벤토리** 파일을 생성합니다.
2. 플레이북 파일(예: `new-cluster.yml`)을 생성합니다.

다음 예제 플레이북 파일은 펜싱, 여러 리소스, 리소스 그룹을 포함하는 클러스터를 구성합니다. 리소스 그룹에 대한 리소스 복제도 포함합니다. 프로덕션용 플레이북 파일을 생성할 때 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 암호를 암호화하는 것이 좋습니다.

```
- hosts: node1 node2
vars:
  ha_cluster_cluster_name: my-new-cluster
  ha_cluster_hacluster_password: password
  ha_cluster_resource_primitives:
    - id: xvm-fencing
      agent: 'stonith:fence_xvm'
      instance_attrs:
        - attrs:
            - name: pcmk_host_list
              value: node1 node2
    - id: simple-resource
      agent: 'ocf:pacemaker:Dummy'
    - id: resource-with-options
      agent: 'ocf:pacemaker:Dummy'
      instance_attrs:
        - attrs:
            - name: fake
              value: fake-value
            - name: passwd
              value: passwd-value
      meta_attrs:
        - attrs:
            - name: target-role
              value: Started
            - name: is-managed
              value: 'true'
      operations:
```

```

- action: start
  attrs:
    - name: timeout
      value: '30s'
- action: monitor
  attrs:
    - name: timeout
      value: '5'
    - name: interval
      value: '1min'
- id: dummy-1
  agent: 'ocf:pacemaker:Dummy'
- id: dummy-2
  agent: 'ocf:pacemaker:Dummy'
- id: dummy-3
  agent: 'ocf:pacemaker:Dummy'
- id: simple-clone
  agent: 'ocf:pacemaker:Dummy'
- id: clone-with-options
  agent: 'ocf:pacemaker:Dummy'
ha_cluster_resource_groups:
- id: simple-group
  resource_ids:
    - dummy-1
    - dummy-2
  meta_attrs:
    - attrs:
        - name: target-role
          value: Started
        - name: is-managed
          value: 'true'
- id: cloned-group
  resource_ids:
    - dummy-3
ha_cluster_resource_clones:
- resource_id: simple-clone
- resource_id: clone-with-options
  promotable: yes
  id: custom-clone-id
  meta_attrs:
    - attrs:
        - name: clone-max
          value: '2'
        - name: clone-node-max
          value: '1'
- resource_id: cloned-group
  promotable: yes

roles:
- rhel-system-roles.ha_cluster

```

3.

파일을 저장합니다.

4.

1단계에서 생성한 인벤토리 파일 *인벤토리*의 경로를 지정하여 플레이북을 실행합니다.

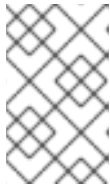
```
# ansible-playbook -i inventory new-cluster.yml
```

22.5. 리소스 제약 조건을 사용하여 고가용성 클러스터 구성

다음 절차에서는 **ha_cluster** 시스템 역할을 사용하여 리소스 위치 제한 조건, 리소스 공동 배치 제한 조건, 리소스 순서 제약 조건, 리소스 티켓 제약 조건을 포함하는 고가용성 클러스터를 생성합니다.

사전 요구 사항

- 플레이북을 실행할 노드에 **ansible-core**가 설치되어 있어야 합니다.



참고

ansible-core를 클러스터 멤버 노드에 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

RHEL 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#)를 참조하십시오.

- 클러스터 구성원으로 사용할 **RHEL**을 실행하는 시스템에는 **RHEL** 및 **RHEL** 고가용성 애드온에 대해 유효한 서브스크립션 적용 범위가 있어야 합니다.



주의

ha_cluster 시스템 역할은 지정된 노드의 기존 클러스터 구성을 대체합니다. 역할에 지정되지 않은 설정은 모두 손실됩니다.

절차

1.

ha_cluster 시스템 역할에 대한 인벤토리 지정에 설명된 대로 클러스터에 노드를 지정하는 인

벤토리 파일을 생성합니다.

2.

플레이북 파일(예: **new-cluster.yml**)을 생성합니다.

다음 예제 플레이북 파일은 리소스 위치 제한 조건, 리소스 위치 제한 조건, 리소스 순서 제약 조건 및 리소스 티켓 제약 조건을 포함하는 클러스터를 구성합니다. 프로덕션용 플레이북 파일을 생성할 때 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 암호를 암호화하는 것이 좋습니다.

```
- hosts: node1 node2
vars:
  ha_cluster_cluster_name: my-new-cluster
  ha_cluster_hacluster_password: password
  # In order to use constraints, we need resources the constraints will apply
  # to.
  ha_cluster_resource_primitives:
    - id: xvm-fencing
      agent: 'stonith:fence_xvm'
      instance_attrs:
        - attrs:
            - name: pcmk_host_list
              value: node1 node2
    - id: dummy-1
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-2
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-3
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-4
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-5
      agent: 'ocf:pacemaker:Dummy'
    - id: dummy-6
      agent: 'ocf:pacemaker:Dummy'
  # location constraints
  ha_cluster_constraints_location:
    # resource ID and node name
    - resource:
        id: dummy-1
        node: node1
        options:
          - name: score
            value: 20
    # resource pattern and node name
    - resource:
        pattern: dummy-\d+
        node: node1
        options:
          - name: score
            value: 10
    # resource ID and rule
    - resource:
```

```

    id: dummy-2
    rule: '#uname eq node2 and date in_range 2022-01-01 to 2022-02-28'
# resource pattern and rule
- resource:
    pattern: dummy-\d+
    rule: node-type eq weekend and date-spec weekdays=6-7
# colocation constraints
ha_cluster_constraints_colocation:
# simple constraint
- resource_leader:
    id: dummy-3
  resource_follower:
    id: dummy-4
  options:
    - name: score
      value: -5
# set constraint
- resource_sets:
    - resource_ids:
        - dummy-1
        - dummy-2
    - resource_ids:
        - dummy-5
        - dummy-6
    options:
        - name: sequential
          value: "false"
  options:
    - name: score
      value: 20
# order constraints
ha_cluster_constraints_order:
# simple constraint
- resource_first:
    id: dummy-1
  resource_then:
    id: dummy-6
  options:
    - name: symmetrical
      value: "false"
# set constraint
- resource_sets:
    - resource_ids:
        - dummy-1
        - dummy-2
    options:
        - name: require-all
          value: "false"
        - name: sequential
          value: "false"
    - resource_ids:
        - dummy-3
    - resource_ids:
        - dummy-4
        - dummy-5
    options:

```

```

    - name: sequential
      value: "false"
# ticket constraints
ha_cluster_constraints_ticket:
  # simple constraint
  - resource:
      id: dummy-1
      ticket: ticket1
      options:
        - name: loss-policy
          value: stop
  # set constraint
  - resource_sets:
      - resource_ids:
          - dummy-3
          - dummy-4
          - dummy-5
      ticket: ticket2
      options:
        - name: loss-policy
          value: fence

roles:
  - linux-system-roles.ha_cluster

```

3.

파일을 저장합니다.

4.

1단계에서 생성한 인벤토리 파일 *인벤토리*의 경로를 지정하여 플레이북을 실행합니다.

```
# ansible-playbook -i inventory new-cluster.yml
```

22.6. HA 클러스터 시스템 역할을 사용하여 고가용성 클러스터에서 APACHE HTTP 서버 구성

이 절차에서는 HA 클러스터 시스템 역할을 사용하여 2-노드 Red Hat Enterprise Linux High Availability Add-On 클러스터에서 활성/수동 Apache HTTP 서버를 구성합니다.

사전 요구 사항

•

플레이북을 실행할 노드에 **ansible-core**가 설치되어 있어야 합니다.



참고

ansible-core를 클러스터 멤버 노드에 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **RHEL** 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 [RHEL 시스템 역할 시작하기](#) 를 참조하십시오.
- 클러스터 구성원으로 사용할 **RHEL**을 실행하는 시스템에는 **RHEL** 및 **RHEL** 고가용성 애드온에 대해 유효한 서브스크립션 적용 범위가 있어야 합니다.
- 시스템에 **Apache**에 필요한 공용 가상 **IP** 주소가 포함되어 있습니다.
- 시스템에는 **iSCSI**, 파이버 채널 또는 기타 공유 네트워크 블록 장치를 사용하여 클러스터의 노드를 위한 공유 스토리지가 포함됩니다.
- **Pacemaker** 클러스터에서 **ext4** 파일 시스템을 사용하여 **LVM** 볼륨 구성에 설명된 대로 **ext4** 파일 시스템으로 **LVM** 논리 볼륨을 구성했습니다.
- **Apache HTTP** 서버 구성에 설명된 대로 **Apache HTTP** 서버를 구성했습니다.
- 시스템에는 클러스터 노드를 펜싱하는 데 사용할 **APC** 전원 스위치가 포함됩니다.



주의

HA Cluster System Role은 지정된 노드의 기존 클러스터 구성을 대체합니다. 역할에 지정되지 않은 설정은 모두 손실됩니다.

절차

1. **HA** 클러스터 시스템 역할에 대한 인벤토리 지정에 설명된 대로 클러스터에 노드를 지정하는 인벤토리 파일을 생성합니다.

2.

플레이북 파일(예: **http-cluster.yml**)을 생성합니다.

다음 예제 플레이북 파일은 활성/수동 두 노드 **HA** 클러스터에서 이전에 생성된 **Apache HTTP** 서버를 구성합니다.

이 예에서는 호스트 이름이 **z1pc.example.com** 인 **APC** 전원 스위치를 사용합니다. 클러스터에서 다른 펜스 에이전트를 사용하지 않는 경우 이 예와 같이 **ha_cluster_fence_agent_packages** 변수를 정의할 때 클러스터에 필요한 펜스 에이전트만 나열할 수 있습니다.

프로덕션용 플레이북 파일을 생성할 때 **Ansible Vault**를 사용하여 콘텐츠 암호화에 설명된 대로 암호를 암호화하는 것이 좋습니다.

```
- hosts: z1.example.com z2.example.com
roles:
  - rhel-system-roles.ha_cluster
vars:
  ha_cluster_hacluster_password: password
  ha_cluster_cluster_name: my_cluster
  ha_cluster_fence_agent_packages:
    - fence-agents-apc-snmp
  ha_cluster_resource_primitives:
    - id: myapc
      agent: stonith:fence_apc_snmp
      instance_attrs:
        - attrs:
            - name: ipaddr
              value: z1pc.example.com
            - name: pcmk_host_map
              value: z1.example.com:1;z2.example.com:2
            - name: login
              value: apc
            - name: passwd
              value: apc
    - id: my_lvm
      agent: ocf:heartbeat:LVM-activate
      instance_attrs:
        - attrs:
            - name: vgname
              value: my_vg
            - name: vg_access_mode
              value: system_id
    - id: my_fs
      agent: Filesystem
      instance_attrs:
        - attrs:
            - name: device
              value: /dev/my_vg/my_lv
            - name: directory
```

```

        value: /var/www
      - name: fstype
        value: ext4
    - id: VirtualIP
      agent: IPAddr2
      instance_attrs:
        - attrs:
            - name: ip
              value: 198.51.100.3
            - name: cidr_netmask
              value: 24
    - id: Website
      agent: apache
      instance_attrs:
        - attrs:
            - name: configfile
              value: /etc/httpd/conf/httpd.conf
            - name: statusurl
              value: http://127.0.0.1/server-status
  ha_cluster_resource_groups:
    - id: apachegroup
      resource_ids:
        - my_lvm
        - my_fs
        - VirtualIP
        - Website

```

3.

파일을 저장합니다.

4.

1단계에서 생성한 인벤토리 파일 *인벤토리*의 경로를 지정하여 플레이북을 실행합니다.

```
# ansible-playbook -i inventory http-cluster.yml
```

검증 단계

1.

클러스터의 노드 중 하나에서 클러스터 상태를 확인합니다. 4개의 모든 리소스는 모두 동일한 노드 **z1.example.com**에서 실행됩니다.

구성한 리소스가 실행 중이지 않음을 발견하면 **pcs resource debug-start resource** 명령을 실행하여 리소스 구성을 테스트할 수 있습니다.

```

[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 16:38:51 2013
Last change: Wed Jul 31 16:42:14 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687

```

```
2 Nodes configured
6 Resources configured
```

```
Online: [ z1.example.com z2.example.com ]
```

```
Full list of resources:
```

```
myapc (stonith:fence_apc_snmp):    Started z1.example.com
Resource Group: apachegroup
  my_lvm (ocf::heartbeat:LVM-activate): Started z1.example.com
  my_fs (ocf::heartbeat:Filesystem): Started z1.example.com
  VirtualIP (ocf::heartbeat:IPaddr2): Started z1.example.com
  Website (ocf::heartbeat:apache): Started z1.example.com
```

2.

클러스터가 가동되어 실행되면 브라우저에서 **IPaddr2** 리소스로 정의된 **IP** 주소를 가리키면 간단한 단어 "Hello"로 구성된 샘플 디스플레이를 볼 수 있습니다.

```
Hello
```

3.

z1.example.com 에서 실행 중인 리소스 그룹이 **z2.example.com** 노드에 실패했는지 여부를 테스트하려면 노드 **z1.example.com** 을 대기 모드로 배치한 후 노드가 더 이상 리소스를 호스트할 수 없습니다.

```
[root@z1 ~]# pcs node standby z1.example.com
```

4.

노드 **z1** 을 **standby** 모드로 전환한 후 클러스터의 노드 중 하나에서 클러스터 상태를 확인합니다. 이제 리소스는 모두 **z2** 에서 실행 중이어야 합니다.

```
[root@z1 ~]# pcs status
Cluster name: my_cluster
Last updated: Wed Jul 31 17:16:17 2013
Last change: Wed Jul 31 17:18:34 2013 via crm_attribute on z1.example.com
Stack: corosync
Current DC: z2.example.com (2) - partition with quorum
Version: 1.1.10-5.el7-9abe687
2 Nodes configured
6 Resources configured
```

```
Node z1.example.com (1): standby
Online: [ z2.example.com ]
```

```
Full list of resources:
```

```
myapc (stonith:fence_apc_snmp):    Started z1.example.com
Resource Group: apachegroup
  my_lvm (ocf::heartbeat:LVM-activate): Started z2.example.com
  my_fs (ocf::heartbeat:Filesystem): Started z2.example.com
  VirtualIP (ocf::heartbeat:IPaddr2): Started z2.example.com
  Website (ocf::heartbeat:apache): Started z2.example.com
```

정의된 IP 주소의 웹 사이트는 중단 없이 계속 표시되어야 합니다.

5.

대기 모드에서 **z1** 을 제거하려면 다음 명령을 입력합니다.

```
[root@z1 ~]# pcs node unstandby z1.example.com
```



참고

대기 모드에서 노드를 제거하면 리소스가 해당 노드로 다시 실패하지 않습니다. 이는 리소스의 **resource-stickiness** 값에 따라 달라집니다. **resource-stickiness meta** 속성에 대한 자세한 내용은 [현재 노드를 선호하도록 리소스 구성](#) 을 참조하십시오.

22.7. 추가 리소스

- [RHEL 시스템 역할 시작하기](#)
- `/usr/share/ansible/roles/ rhel-system-roles.logging/README.html`의 `rhel-system-roles` 패키지과 함께 설치된 설명서
- [RHEL 시스템 역할 KB 문서](#)
- `ansible-playbook(1)` 도움말 페이지.

23장. COCKPIT RHEL 시스템 역할을 사용하여 웹 콘솔 설치 및 구성

cockpit RHEL System Role을 사용하면 시스템에 웹 콘솔을 설치하고 구성할 수 있습니다.

23.1. COCKPIT 시스템 역할

cockpit 시스템 역할을 사용하여 웹 콘솔을 자동으로 배포 및 활성화하여 웹 브라우저에서 **RHEL** 시스템을 관리할 수 있습니다.

23.2. COCKPIT RHEL 시스템 역할에 대한 변수

cockpit RHEL 시스템 역할에 사용되는 매개변수는 다음과 같습니다.

역할 변수	설명
cockpit_packages: (기본값: default)	미리 정의된 패키지 세트(default, minimal 또는 full) 중 하나를 설정합니다. * cockpit_packages: (기본값: default) - 대부분의 일반 페이지 및 온디맨드 설치 UI * cockpit_packages: (기본값: 최소) - 개요, 터미널, 로그, 계정 및 지표 페이지만, 최소한의 종속 항목 * cockpit_packages: (기본값: full) - 사용 가능한 모든 페이지 원하는 경우 설치할 cockpit 패키지 고유의 선택 사항을 지정합니다.
cockpit_enabled: (default:yes)	부팅 시 웹 콘솔 웹 서버가 자동으로 시작되도록 설정되어 있는지 구성
cockpit_started: (default:yes)	웹 콘솔을 시작해야 하는 경우 구성
cockpit_config: (기본값: 없음)	/etc/cockpit/cockpit.conf 파일에 설정을 적용할 수 있습니다. 알림: 이전 설정 파일이 손실됩니다.

추가 리소스

- **/usr/share/ansible/rhel-system-roles/cockpit/README.md** 파일.

- [Cockpit 구성 파일](#) 도움말 페이지.

23.3. COCKPIT RHEL 시스템 역할을 사용하여 웹 콘솔 설치

다음 단계에 따라 시스템에 웹 콘솔을 설치하고 해당 콘솔에서 서비스에 액세스할 수 있도록 합니다.

사전 요구 사항

- VPN 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- Red Hat Ansible Core가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 내용으로 새 **playbook.yml** 파일을 생성합니다.

```
---
- hosts: all
  tasks:
    - name: Install RHEL web console
      include_role:
        name: rhel-system-roles.cockpit
      vars:
        cockpit_packages: default
        #cockpit_packages: minimal
        #cockpit_packages: full

    - name: Configure Firewall for web console
      include_role:
        name: rhel-system-roles.firewall
      vars:
        firewall:
          service: cockpit
          state: enabled
```



참고

cockpit 포트는 **firewalld**에서 기본적으로 열려 있으므로 시스템 관리자가 이를 사용자 지정한 경우에만 "웹 콘솔 구성" 작업이 적용됩니다.

2.

선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check -i inventory_file playbook.yml
```

3.

인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook.yml
```

추가 리소스

- [웹 콘솔 설치 및 활성화.](#)

23.4. 인증서 RHEL 시스템 역할을 사용하여 새 인증서 설정

기본적으로 웹 콘솔은 첫 번째 시작 시 자체 서명된 인증서를 생성합니다. 보안상의 이유로 자체 서명된 인증서를 사용자 지정할 수 있습니다. 새 인증서를 생성하려면 인증서 역할을 사용할 수 있습니다. 이를 위해 다음 단계를 수행합니다.

사전 요구 사항

- **VPN** 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형* 노드에 대한 액세스 및 권한.
- **Red Hat Ansible Core**가 기타 시스템을 구성하는 시스템인 *제어* 노드 액세스 및 사용 권한.

제어 노드에서 다음이 있어야 합니다.

- **ansible-core** 및 **rhel-system-roles** 패키지가 설치됩니다.
- 관리 노드를 나열하는 인벤토리 파일.

절차

1. 다음 콘텐츠를 사용하여 새 **playbook2.yml** 파일을 생성합니다.

```
---
- hosts: all
  tasks:
    - name: Generate Cockpit web server certificate
      include_role:
        name: rhel-system-roles.certificate
      vars:
        certificate_requests:
          - name: /etc/cockpit/ws-certs.d/01-certificate
            dns: ['localhost', 'www.example.com']
            ca: ipa
            group: cockpit-ws
```

2. 선택 사항: 플레이북 구문을 확인합니다.

```
# ansible-playbook --syntax-check -i inventory_file playbook2.yml
```

3. 인벤토리 파일에서 플레이북을 실행합니다.

```
# ansible-playbook -i inventory_file /path/to/file/playbook2.yml
```

추가 리소스

- [RHEL 시스템 역할을 사용하여 인증서 요청.](#)