# Red Hat Enterprise Linux 7

## RHEL 7.9의 시스템 역할을 사용하는 관리 및 구성 작업

Red Hat Ansible Automation Platform 플레이북을 사용하여 RHEL 시스템 역할 적용을 통해 시스템 관리 작업 수행

Last Updated: 2023-07-14

# Red Hat Enterprise Linux 7 RHEL 7.9의 시스템 역할을 사용하는 관리 및 구성 작업

Red Hat Ansible Automation Platform 플레이북을 사용하여 RHEL 시스템 역할 적용을 통해 시스템 관리 작업 수행

## 법적 공지



## 초록

RHEL 7.9에서 시스템 관리 작업을 수행하기 위해 Red Hat Ansible Automation Platform 플레이북을 사용하여 RHEL 시스템 역할 적용

# 차례

# 오픈 소스를 더 포괄적으로 만들기

Red Hat은 코드, 문서 및 웹 속성에서 문제가 있는 언어를 교체하기 위해 최선을 다하고 있습니다. 마스터, 슬레이브, 블랙리스트, 화이트리스트라는 네 가지 용어로 시작하고 있습니다. 이러한 변경 사항은 향후 여러 릴리스에 대해 점차 구현될 예정입니다. 자세한 내용은 CTO Chris Wright의 메시지에서 참조하십시오.

# RED HAT 문서에 관한 피드백 제공

문서 개선을 위한 의견에 감사드립니다. 어떻게 개선할 수 있는지 알려주십시오.

특정 문구에 대한 의견 제출

1. **Multi-page HTML** 형식으로 설명서를 보고 페이지가 완전히 로드된 후 오른쪽 상단 모서리에 피드백 버튼이 표시되는지 확인합니다.
2. 커서를 사용하여 주석 처리할 텍스트 부분을 강조 표시합니다.
3. 강조 표시된 텍스트 옆에 표시되는 피드백 추가 버튼을 클릭합니다.
4. 의견을 추가하고 제출을 클릭합니다.

**Bugzilla**를 통해 피드백 제출(등록 필요)

1. Bugzilla 웹 사이트에 로그인합니다.
2. 버전 메뉴에서 올바른 버전을 선택합니다.
3. 요약 필드에 설명 제목을 입력합니다.
4. 설명 필드에 개선을 위한 제안을 입력합니다. 문서의 관련 부분에 대한 링크를 포함합니다.
5. **Submit Bug**를 클릭하십시오.

# 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 새 주요 릴리스를 채택할 수 있습니다.

{RHEL7.9}에서 인터페이스는 현재 다음 역할로 구성됩니다.

- **certificate**
- **crypto_policy**
- **kdump**
- **kernel_settings**
- **logging**
- 메트릭
- **nbde_client** 및 **nbde_server**
- **network**
- **selinux**
- **ssh**
- **sshd**
- **storage**
- **timesync**
- **tlog**

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

관련 자료

- RHEL (Red Hat Enterprise Linux) 시스템 역할
- **usr/share/doc/rhel-system-roles-VERSION** 문서 [1]
- **selinux** 시스템 역할 소개
- 스토리지 역할 소개

## 1.2. RHEL 시스템 역할 용어

이 문서에서는 다음 용어를 찾을 수 있습니다.

시스템 역할 용어

**Ansible Playbook**

플레이북은 Ansible의 구성, 배포 및 오케스트레이션 언어입니다. 원격 시스템이 강제 적용하려는 정책 또는 일반적인 IT 프로세스의 단계 집합을 설명할 수 있습니다.

제어 노드

Ansible이 설치된 모든 시스템. 모든 제어 노드에서 /usr/bin/ansible 또는 /usr/bin/ansible-playbook을 호출하여 명령과 플레이북을 실행할 수 있습니다. Python이 제어 노드로 설치된 컴퓨터를 사용할 수 있습니다. 노트북, 공유 데스크탑 및 서버는 모두 Ansible을 실행할 수 있습니다. 그러나 Windows 머신을 제어 노드로 사용할 수는 없습니다. 여러 제어 노드가 있을 수 있습니다.

인벤토리

관리형 노드 목록. 인벤토리 파일을 "hostfile"이라고도 합니다. 인벤토리는 각 관리 노드의 IP 주소와 같은 정보를 지정할 수 있습니다. 인벤토리는 쉽게 확장하기 위해 관리 노드, 생성 및 중첩 그룹을 구성할 수도 있습니다. 인벤토리에 대한 자세한 내용은 인벤토리 작업 섹션을 참조하십시오.

관리형 노드

Ansible을 사용하여 관리하는 네트워크 장치, 서버 또는 둘 다입니다. 관리되는 노드를 "호스트"라고도 합니다. Ansible은 관리 노드에 설치되지 않습니다.

//// 다음에 포함된 모듈:

## 1.3. 역할 적용

다음 절차에서는 특정 역할을 적용하는 방법을 설명합니다.

사전 요구 사항

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

```
# yum install rhel-system-roles
```

- RHEL 시스템 역할을 사용하는 플레이북을 실행하려면 **ansible** 패키지가 필요합니다. Ansible Engine 리포지토리가 활성화되어 있고 **ansible** 패키지가 제어 노드로 사용하려는 시스템에 설치되어 있는지 확인합니다.
  - Red Hat Ansible Engine 서브스크립션이 없는 경우 Red Hat Enterprise Linux 서브스크립션과 함께 제공되는 제한된 지원 버전의 Red Hat Ansible Engine을 사용할 수 있습니다. 이 경우 다음 단계를 따르십시오.
    1. RHEL Ansible Engine 리포지토리를 활성화합니다.

```
# subscription-manager refresh
# sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

    2. Ansible Engine을 설치합니다.

```
# yum install ansible
```

  - Red Hat Ansible Engine 서브스크립션이 있는 경우 How do I download and Install Red Hat Ansible Engine? 에 설명된 절차를 따르십시오.

- Ansible 인벤토리를 생성할 수 있는지 확인합니다.
  인벤토리는 Ansible 플레이북에서 사용하는 호스트, 호스트 그룹 및 일부 구성 매개 변수를 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 개의 원격 시스템을 배포하거나 수동 주문 프로세스의 단계를 오케스트레이션할 수 있습니다.

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

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

절차

1. 관리할 호스트 및 그룹을 포함하는 필수 Ansible 인벤토리를 생성합니다. 다음은 **webservers**:이라는 호스트 그룹의 **inventory.ini** 파일을 사용하는 예입니다.

```
[webservers]
host1
host2
host3
```

2. 필요한 역할을 포함하여 Ansible 플레이북을 생성합니다. 다음 예는 플레이북에 **roles:** 옵션을 통해 역할을 사용하는 방법을 보여줍니다.
   다음 예제는 지정된 플레이에 대한 **roles:** 옵션을 통해 역할을 사용하는 방법을 보여줍니다.

```
---
- hosts: webservers
  roles:
    - rhel-system-roles.network
    - rhel-system-roles.timesync
```

참고

모든 역할에는 역할 사용 방법과 지원되는 매개 변수 값을 설명하는 README 파일이 포함되어 있습니다. 역할의 설명서 디렉터리에서 특정 역할에 대한 플레이북 예제도 찾을 수 있습니다. 이러한 문서 디렉터리는 **rhel-system-roles** 패키지와 함께 기본적으로 제공되며 다음 위치에서 찾을 수 있습니다.

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

**rhel-7.9 rhel-system-roles** 패키지의 경우 *VERSION* 값은 **1.0.1** 입니다.

*SUBSYSTEM* 을 **selinux**,**kdump**,**network**,**timesync** 또는 **storage** 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** 및 명령을 실행합니다.

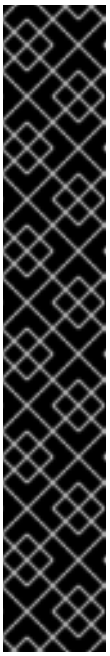
```
# ansible-playbook name.of.the.playbook
```

- 인벤토리를 편집하여 사용할 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. 여러 대상 호스트가 있지만 플레이북을 실행할 호스트를 선택하려는 경우 플레이북에 변수를 추가하여 호스트를 선택할 수 있습니다. 예를 들어 다음과 같습니다.

```
Ansible Playbook | example-playbook.yml:

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

플레이북 실행 명령:

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

관련 자료

- Ansible 플레이북
- Ansible 플레이북에서 역할 사용
- Ansible 플레이북의 예
- 인벤토리를 만들고 작업하는 방법은 무엇입니까?
- ansible-playbook

## 1.4. 관련 자료

- Red Hat Enterprise Linux RHEL System Roles Red Hat Knowledgebase 문서
- 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 필수 패키지를 설치합니다.

사전 요구 사항

- Red Hat Ansible Engine 서브스크립션이 있어야 합니다. Red Hat Ansible Engine을 다운로드 및 설치하려면 어떻게 해야 합니까?
- 제어 노드로 사용할 시스템에 Ansible 패키지가 설치되어 있어야 합니다.

절차

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

```
# yum install rhel-system-roles
```

Red Hat Ansible Engine 서브스크립션이 없는 경우 Red Hat Enterprise Linux 서브스크립션과 함께 제공되는 제한된 지원 버전의 Red Hat Ansible Engine을 사용할 수 있습니다. 이 경우 다음 단계를 따르십시오.

a. RHEL Ansible Engine 리포지토리를 활성화합니다.

```
# subscription-manager refresh

# sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

b. Ansible Engine을 설치합니다.

```
# yum install ansible
```

결과적으로 Ansible 플레이북을 생성할 수 있습니다.

관련 자료

- RHEL 시스템 역할 개요는 RHEL (Red Hat Enterprise Linux) 시스템 역할을 참조하십시오.
- ansible-playbook 명령 사용에 대한 자세한 내용은 ansible-playbook 도움말 페이지를 참조하십시오.

# 3장. 컬렉션 설치 및 사용

## 3.1. ANSIBLE 컬렉션 소개

Ansible 컬렉션은 자동화를 배포, 유지 관리 및 사용하는 새로운 방법입니다. 플레이북, 역할, 모듈 및 플러그인과 같은 여러 유형의 Ansible 콘텐츠를 결합하면 유연성과 확장성 면에서 이점을 얻을 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 컬렉션이 네임스페이스 와 컬렉션 이름으로 구성된 정규화된 컬렉션 이름 (FN)의 개념을 사용한다는 점입니다 **.** 사용하는 네임스페이스 는 **redhat** 이며 컬렉션 이름은 **rhel_system_roles** 입니다. 따라서 **kernel_settings** 역할의 기존 RHEL 시스템 역할 형식은 **rhel-system-roles.kernel_settings** 로 표시되지만 **kernel_settings** 역할의 Collection 정규화된 컬렉션 이름을 사용하여 **redhat.rhel_system_roles.kernel_settings** 로 표시됩니다.

네임스페이스 와 컬렉션 이름을 조합하면 오브젝트가 고유합니다. 또한 충돌 없이 Ansible 컬렉션 및 네임스페이스 전반에서 오브젝트를 공유할 수 있습니다.

관련 자료

- Red Hat Certified Collections는 자동화 허브에 액세스하여 찾을 수 있습니다.

## 3.2. 컬렉션 구조

컬렉션은 Ansible 콘텐츠의 패키지 형식입니다. 데이터 구조는 다음과 같습니다.

- Documentation/: 역할이 문서를 제공하는 경우 컬렉션에 대한 로컬 문서(예 포함)
- Galaxy.yml: Ansible 컬렉션 패키지에 포함될 MANIFEST.json의 소스 데이터
- playbooks/: 플레이북을 사용할 수 있습니다.
  - tasks/: include_tasks/import_tasks 사용을 위한 'task list files'를 포함합니다.
- plugins/: 모든 Ansible 플러그인 및 모듈을 여기에서 사용할 수 있으며 각 하위 디렉터리에 있습니다.
  - modules/: Ansible 모듈
  - modules_utils/: 모듈을 개발하기 위한 공통 코드
  - lookup/: 플러그인 검색
  - 필터/: Jinja2 필터 플러그인
  - connection/: 기본값을 사용하지 않는 경우 연결 플러그인 필요
- roles/: Ansible 역할의 디렉터리
- tests/: 컬렉션 내용 테스트

## 3.3. CLI를 사용하여 컬렉션 설치

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

Automation Hub를 통해 브라우저 또는 명령줄을 사용하여 컬렉션을 설치할 수 있습니다.

사전 요구 사항

- Red Hat Ansible Engine 버전 2.9 이상 설치됨.
- **python3-jmespath** 패키지가 설치되어 있습니다.
- 관리 노드를 나열하는 인벤토리 파일이 있습니다.

절차

- RPM 패키지를 통해 컬렉션을 설치합니다.

```
# yum install rhel-system-roles
```

설치가 완료되면 역할은 **redhat.rhel_system_roles.<role_name>** 으로 사용할 수 있습니다. 또한 **/usr/share/doc/rhel-system-roles-*VERSION*/collection/README.md** 및 **/ usr/share/doc/rhel-system-roles-*VERSION*/collection/roles/<role_name>/README.md** 에서 각 역할에 대한 설명서를 찾을 수 있습니다.

검증 단계

컬렉션이 성공적으로 설치되었는지 확인하려면 localhost에 kernel_settings를 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. 파일을 편집하여 "hosts: all"을 "hosts: localhost"로 대체하여 플레이북이 로컬 시스템에서만 실행되도록 합니다.
3. 점검 모드에서 ansible-playbook을 실행합니다. 이렇게 하면 시스템의 설정이 변경되지 않습니다.

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

이 명령은 **failed=0** 값을 반환합니다.

관련 자료

- ansible-playbook 도움말 페이지를 참조하십시오.

## 3.4. AUTOMATION HUB에서 컬렉션 설치

Automation Hub를 사용하는 경우 Automation Hub에 호스팅되는 시스템 역할 컬렉션을 설치할 수 있습니다.

사전 요구 사항

- Red Hat Ansible Engine 버전 2.9 이상이 설치되어 있습니다.
- **python3-jmespath** 패키지가 설치되어 있습니다.

- 관리 노드를 나열하는 인벤토리 파일이 있습니다.

절차

1. Automation Hub에서 **redhat.rhel_system_roles** 컬렉션을 설치합니다.

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

2. Red Hat Automation Hub를 **ansible.cfg** 구성 파일의 기본 콘텐츠 소스로 정의합니다. 콘텐츠의 기본 소스로 Red Hat Automation Hub 구성을 참조하십시오.
설치가 완료되면 역할은 **redhat.rhel_system_roles.<role_name>** 으로 사용할 수 있습니다. 또한 **/usr/share/ansible/collections/ansible_collections/redhat/rhel_system_roles/roles/<role_name>/README.md** 에서 각 역할에 대한 설명서를 찾을 수 있습니다.

검증 단계

컬렉션이 성공적으로 설치되었는지 확인하려면 localhost에 kernel_settings를 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. 파일을 편집하여 "hosts: all"을 "hosts: localhost"로 대체하여 플레이북이 로컬 시스템에서만 실행되도록 합니다.

3. 점검 모드에서 ansible-playbook을 실행합니다. 이렇게 하면 시스템의 설정이 변경되지 않습니다.

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

명령이 **failed=0** 값으로 반환되는 것을 확인할 수 있습니다.

관련 자료

- ansible-playbook 도움말 페이지를 참조하십시오.

## 3.5. 컬렉션을 사용하여 로컬 로깅 시스템 역할 적용

다음은 Collections를 사용하여 Red Hat Ansible Engine 플레이북을 준비하고 적용하여 별도의 시스템 세트에서 로깅 솔루션을 구성하는 예입니다.

사전 요구 사항

- **rhel-system-roles** 의 컬렉션 형식은 **rpm** 패키지 또는 자동화 허브에서 설치됩니다.

절차

1. 필요한 역할을 정의하는 플레이북을 생성합니다.

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

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

b. YAML 파일에 다음 내용을 삽입합니다.

```
---
- name: Deploying basics input and implicit files output
  hosts: all
  roles:
    - redhat.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 logging-playbook.yml
```

다음과 같습니다.

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

검증 단계

1. **/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. 시스템이 로그에 메시지를 전송하는지 확인합니다.

a. 테스트 메시지를 전송합니다.

```
# 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
```

**hostname** 은 클라이언트 시스템의 호스트 이름입니다. 로그에 logger 명령을 입력한 사용자의 사용자 이름(이 경우 **root** )이 표시됩니다.

# 4장. ANSIBLE 역할을 사용하여 커널 매개 변수 영구 구성

Red Hat Ansible Engine에 대한 지식이 있는 숙련된 사용자는 **kernel_settings** 역할을 사용하여 여러 클라이언트에 대한 커널 매개 변수를 한 번에 구성할 수 있습니다. 이 솔루션은 다음과 같습니다.

- 효율적인 입력 설정을 갖춘 친숙한 인터페이스를 제공합니다.
- 의도한 모든 커널 매개 변수를 한 위치에 유지합니다.

제어 머신에서 **kernel_settings** 역할을 실행하면 커널 매개 변수가 관리 시스템에 즉시 적용되고 재부팅 시 지속됩니다.

## 4.1. 커널 설정 역할 소개

RHEL 시스템 역할은 여러 시스템을 원격으로 관리하는 일관된 구성 인터페이스를 제공하는 Ansible Automation Platform의 역할 및 모듈 컬렉션입니다.

**kernel_settings** 시스템 역할을 사용하여 커널의 자동화된 구성을 위해 RHEL 시스템 역할이 도입되었습니다. **rhel-system-roles** 패키지에는 이 시스템 역할 및 참조 문서가 포함되어 있습니다.

자동화된 방식으로 하나 이상의 시스템에서 커널 매개 변수를 적용하려면 플레이북에서 선택한 역할 변수 중 하나 이상과 함께 **kernel_settings** 역할을 사용합니다. 플레이북은 사람이 읽을 수 있으며 YAML 형식으로 작성된 하나 이상의 플레이 목록입니다.

인벤토리 파일을 사용하여 Ansible Engine에서 플레이북에 따라 구성할 시스템 집합을 정의할 수 있습니다.

**kernel_settings** 역할을 사용하면 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
- **systemd** 서비스 관리자의 CPU 선호도 및 **kernel_settings_systemd_cpu_affinity** 역할 변수를 사용하여 포크 처리
- 커널 메모리 하위 시스템은 kernel_settings_transparent **_hugepages** 및 **kernel_settings_transparent_hugepages _defrag** 역할 변수를 사용하여 hugepages를 투명하게 합니다.

관련 자료

- **/usr/share/doc/rhel-system-roles-VERSION/kernel_settings/** 디렉터리의 **README.md** 파일
- 플레이북 작업
- 재고를 구축하는 방법

## 4.2. 커널 설정 역할을 사용하여 선택한 커널 매개변수 적용

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

사전 요구 사항

- Red Hat Ansible Engine 서브스크립션이 시스템에 연결되어 있으며, 이를 통해 **kernel_settings** 역할을 실행하려는 *제어 머신* 이라고도 합니다. 자세한 내용은 How do I download and install Red Hat Ansible Engine 문서를 참조하십시오.
- Ansible Engine 리포지토리는 제어 시스템에서 활성화됩니다.
- Ansible Engine은 제어 시스템에 설치됩니다.

참고

커널 매개 변수를 구성하려는 *관리 대상 호스트* 라고도 하는 시스템에 Ansible Engine을 설치할 필요가 없습니다.

- **rhel-system-roles** 패키지는 제어 시스템에 설치됩니다.
- 관리 호스트의 인벤토리가 제어 시스템에 있으며 Ansible Engine은 연결할 수 있습니다.

절차

1. 필요한 경우 그림 목적으로 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 Ansible Engine을 특정 시스템 컬렉션에 대해 보다 효율적으로 실행할 수 있습니다.

2. 구성 파일을 생성하여 Ansible Engine 작업에 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 지정된 관리 호스트에서 사용자 권한을 **root**로 전환하도록 정의합니다. 커널 매개 변수를 성공적으로 구성하려면 이 작업이 필요합니다. Ansible 플레이북이 실행되면 사용자 암호를 묻는 메시지가 표시됩니다. 사용자는 관리 호스트에 연결한 후 **sudo** 를 통해 자동으로 **root** 로 전환합니다.

3. **kernel_settings** 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 정렬된 작업 목록을 포함합니다.

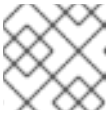
   b. 파일에 다음 내용을 삽입합니다.

   ```
   ---
   - name: Configure kernel settings
     hosts: testingservers

     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

     roles:
       - rhel-system-roles.kernel_settings
   ```

   **name** 키는 선택 사항입니다. 임의의 문자열과 플레이를 레이블로 연결하고 플레이의 용도를 식별합니다. 플레이의 **hosts** 키는 플레이를 실행할 호스트를 지정합니다. 이 키의 값 또는 값은 관리 호스트의 개별 이름으로 제공되거나 인벤토리 파일에 정의된 호스트 그룹으로 제공할 수 있습니다.

   **vars** 섹션은 선택한 커널 매개 변수 이름과 설정해야 하는 값을 포함하는 변수 목록을 나타냅니다.

   **roles** 키는 **vars** 섹션에 언급된 매개 변수 및 값을 구성하기 위해 수행할 시스템 역할을 지정합니다.

   참고

   필요에 맞게 플레이북에서 커널 매개변수와 해당 값을 수정할 수 있습니다.

4. 필요한 경우 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
```

**Ansible Engine**이 플레이북을 실행하기 전에 암호를 묻는 메시지가 표시되고 관리 호스트의 사용자는 커널 매개 변수를 구성하는 데 필요한 **root** 로 전환할 수 있습니다.

**recap** 섹션에는 플레이가 모든 관리 호스트에 대해 성공적으로 완료**(failed=0)**되고 **4**개의 커널 매개 변수가 적용되었음을 보여줍니다**(changed=4)**.

6. 관리 호스트를 다시 시작하고 영향을 받는 커널 매개변수를 확인하여 변경 사항이 적용되었는지 확인하고 재부팅 후에도 지속되는지 확인합니다.

관련 자료

- RHEL 시스템 역할 시작하기
- **/usr/share/doc/rhel-system-roles-VERSION/kernel_settings/** 디렉터리의 **README.md** 파일
- 인벤토리 작업
- Ansible 구성
- 플레이북 사용
- 변수 사용
- 역할

# 5장. 시스템 역할을 사용하여 네트워크 연결 구성

RHEL의 네트워킹 시스템 역할을 통해 관리자는 Ansible을 사용하여 네트워크 관련 구성 및 관리 작업을 자동화할 수 있습니다.

## 5.1. 인터페이스 이름으로 RHEL 시스템 역할을 사용하여 정적 이더넷 연결 구성

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

- 정적 IPv4 주소 - **192.0.2.1** (**/24** 서브넷 마스크 포함)
- 정적 IPv6 주소 - **/64** 서브넷 마스크가 있는**2001:db8:1::1**
- 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 이 절차를 실행합니다.

사전 요구 사항

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

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/ethernet-static-IP.yml** 플레이북을 만듭니다.

```
---
- 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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

## 5.2. 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 연결이 DHCP 서버에서 이 연결에 대한 IP 설정을 요청합니다. Ansible 제어 노드에서 이 절차를 실행합니다.

사전 요구 사항

- DHCP 서버는 네트워크에서 사용할 수 있습니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

- 호스트는 NetworkManager를 사용하여 네트워크를 구성합니다.

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

## 5.3. 시스템 역할을 사용하여 VLAN 태그 구성

Networking RHEL 시스템 역할을 사용하여 VLAN 태그를 구성할 수 있습니다. 다음 절차에서는 이 이더넷 연결을 사용하는 ID 10 이 있는 이더넷 연결 및 VLAN을 추가하는 방법을 설명합니다. 상위 장치로 VLAN 연결에는 IP, 기본 게이트웨이 및 DNS 구성이 포함됩니다.

환경에 따라 그에 따라 플레이를 조정합니다. 예를 들어 다음과 같습니다.

- VLAN을 본딩과 같은 다른 연결에서 포트로 사용하려면 **ip** 속성을 생략하고 상위 구성에서 IP 구성을 설정합니다.
- VLAN에서 팀, 브리지 또는 본딩 장치를 사용하려면 VLAN에서 사용하는 포트의 **interface_name** 및 **type** 특성을 조정합니다.

사전 요구 사항

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

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/vlan-ethernet.yml** 플레이북을 만듭니다.

```
---
- 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: vlan10
        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 프로필의 상위 특성은 **enp1s0** 장치의 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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

## 5.4. RHEL 시스템 역할을 사용하여 네트워크 브리지 구성

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

참고

**Linux** 브리지의 포트에 없는 브리지에 **IP** 구성을 설정합니다.

사전 요구 사항

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

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

## 5.5. RHEL 시스템 역할을 사용하여 네트워크 본딩 구성

Networking RHEL 시스템 역할을 사용하여 네트워크 본딩을 구성할 수 있습니다. 다음 절차에서는 두 이더넷 장치를 사용하고 IPv4 및 IPv6 주소, 기본 게이트웨이 및 DNS 구성을 설정하는 active-backup 모드로 본딩을 구성하는 방법을 설명합니다.

참고

Linux 브리지의 포트에 없는 브리지에 IP 구성을 설정합니다.

사전 요구 사항

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

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/bond-ethernet.yml** 플레이북을 만듭니다.

```
---
- 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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

- **ansible-playbook(1)** 도움말 페이지

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

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

- 정적 IPv4 주소 - **192.0.2.1** (**/24** 서브넷 마스크 포함)
- 정적 IPv6 주소 - **/64** 서브넷 마스크가 있는**2001:db8:1::1**
- 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 이 절차를 실행합니다.

사전 요구 사항

- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 관리 노드에 적절한**sudo** 권한이 있어야 합니다.
- 네트워크는 802.1X 네트워크 인증을 지원합니다.
- 관리 노드는 NetworkManager를 사용합니다.
- TLS 인증에 필요한 다음 파일은 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 인벤토리되지 않은 경우 이 호스트의 IP 또는 이름을 **/etc/ansible/hosts** Ansible 인벤토리 파일에 추가합니다.

   ```
   node.example.com
   ```

2. 다음 내용으로 **~/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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

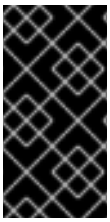
**-u *user_name*** 옵션을 지정하지 않으면 **ansible-playbook** 은 현재 제어 노드에 로그인한 사용자로 관리 호스트에 연결합니다.

관련 자료

- **/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

## 5.7. 시스템 역할을 사용하여 기존 연결에서 기본 게이트웨이 설정

Networking RHEL 시스템 역할을 사용하여 기본 게이트웨이를 설정할 수 있습니다.



중요

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

이미 있는지 여부에 따라 프로시저는 다음 설정을 사용하여 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20 ( /24** 서브넷 마스크 포함)
- 정적 IPv6 주소 - **/64** 서브넷 마스크가 있는 **2001:db8:1::1**
- 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** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.
- 플레이북을 실행할 때 **root** 와 다른 원격 사용자를 사용하는 경우 이 사용자에게는 관리 노드에 대한 적절한 **sudo** 권한이 있습니다.

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

## 5.8. RHEL 시스템 역할을 사용하여 정적 경로 구성

Networking RHEL 시스템 역할을 사용하여 정적 경로를 구성할 수 있습니다.

중요

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

이미 있는지 여부에 따라 프로시저는 다음 설정을 사용하여 **enp7s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20 ( /24** 서브넷 마스크 포함)
- 정적 IPv6 주소 - **/64** 서브넷 마스크가 있는**2001:db8:1::1**
- 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:
  - 게이트웨이가 **198.51.100.1**인 **192.0.2.0/24**
  - 게이트웨이가 **198.51.100.2**인 **203.0.113.0/24**

사전 요구 사항

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

절차

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

```
node.example.com
```

2. 다음 내용으로 **~/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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

   **-u *user_name*** 옵션을 지정하지 않으면 **ansible-playbook** 은 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
  ...
  ```

관련 자료

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

## 5.9. 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 실행할 때 설정이 플레이에 지정된 것과 일치하지 않는 경우 시스템 역할은 이름이 동일한 기존 연결 프로필을 덮어씁니다. 따라서 예를 들어 IP 구성이 이미 있는 경우에도 플레이에서 네트워크 연결 프로필의 전체 구성을 항상 지정합니다. 그렇지 않으면 역할은 이러한 값을 기본값으로 재설정합니다.

이미 있는지 여부에 따라 프로시저는 다음 설정을 사용하여 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 **IPv4** 주소 - **198.51.100.20 ( /24** 서브넷 마스크 포함)
- 정적 **IPv6** 주소 - **/64** 서브넷 마스크가 있는**2001:db8:1::1**
- **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:
  - GRO(Generic receive offload): 비활성화
  - GSO(Generic segmentation offload): 활성화됨
  - TX SCTP(스트림 제어 전송 프로토콜) 분할: 비활성화

사전 요구 사항

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

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 인벤토리되지 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 생성합니다.

```
---
- 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:
            feature:
              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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

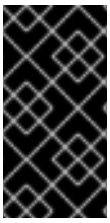
관련 자료

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

- **ansible-playbook(1)** 도움말 페이지

## 5.10. 시스템 역할을 사용하여 ETHTOOL 통합 설정 구성

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



중요

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

이미 있는지 여부에 따라 프로시저는 다음 설정을 사용하여 **enp1s0** 연결 프로필을 생성하거나 업데이트합니다.

- 정적 IPv4 주소 - **198.51.100.20 ( /24** 서브넷 마스크 포함)
- 정적 IPv6 주소 - **/64** 서브넷 마스크가 있는**2001:db8:1::1**
- 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** 통합 설정:
  - RX 프레임: **128**
  - TX 프레임: **128**

사전 요구 사항

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

절차

1. 플레이북에서 명령을 실행할 호스트가 아직 인벤토리되지 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 생성합니다.

```
---
```

```
- 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** 명령이 **-u *user_name*** 옵션에 정의된 사용자의 **sudo** 암호를 입력하라는 메시지를 표시합니다.

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

관련 자료

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

# 6장. 시스템 역할을 사용하여 SELINUX 구성

## 6.1. SELINUX 시스템 역할 소개

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

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

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

표 6.1. **SELinux** 시스템 역할 변수

| 역할 변수 | Description | CLI 대안 |
|---|---|---|
| selinux_policy | 대상 프로세스를 보호하는 정책 선택 또는 다단계 보안 보호. | **/etc/selinux/config**의 **SELINUXTYPE** |
| selinux_state | SELinux 모드를 전환합니다. **ansible-doc selinux**참조 | **/etc/selinux/config** 의 **setenforce** 및 **SELINUX**. |
| selinux_booleans | SELinux 부울 활성화 및 비활성화. **ansible-doc seboolean** 을 참조하십시오. | **setsebool** |
| selinux_fcontexts | SELinux 파일 컨텍스트 매핑 추가 또는 제거. **ansible-doc sefcontext** 를 참조하십시오. | **semanage fcontext** |
| selinux_restore_dirs | 파일 시스템 트리에 SELinux 레이블을 복원합니다. | **restorecon -R** |
| selinux_ports | 포트에 SELinux 레이블을 설정합니다. **ansible-doc seport** 를 참조하십시오. | **semanage** 포트 |
| selinux_logins | 은 사용자를 SELinux 사용자 매핑으로 설정합니다. **ansible-doc selogin** 을 참조하십시오. | **semanage** 로그인 |
| selinux_modules | SELinux 모듈을 설치, 활성화, 비활성화 또는 제거합니다. | **semodule** |

**rhel -system-roles** 패키지에서 설치한 **/usr/share/ansible/roles/rhel-system-roles.selinux/selinux-playbook.yml** 예제 플레이북은 강제 모드로 대상 정책을 설정하는 방법을 보여줍니다. 또한 이 플레이북은 여러 로컬 정책 수정 사항을 적용하고 **/tmp/test_dir/** 디렉터리에 파일 컨텍스트를 복원합니다.

**selinux** 역할 변수에 대한 자세한 참조는**rhel-system-roles** 패키지를 설치하고**/usr/share/doc/rhel-system-roles-*VERSION*/selinux/** 디렉터리에 있는**README.md** 파일을 참조하십시오.

관련 자료

- RHEL 시스템 역할 소개.

## 6.2. SELINUX 시스템 역할을 사용하여 여러 시스템에 SELINUX 설정 적용

단계에 따라 확인된 **SELinux** 설정으로 **Ansible** 플레이북을 준비하고 적용합니다.

사전 요구 사항

- **Red Hat Ansible Engine** 서브스크립션이 시스템에 연결되어 있습니다. 자세한 내용은 How do I download and install Red Hat Ansible Engine 문서를 참조하십시오.

절차

1. **RHEL Ansible** 리포지토리를 활성화합니다. 예를 들면 다음과 같습니다.

```
# sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

2. **Ansible Engine**을 설치합니다.

```
# yum install ansible
```

3. **RHEL** 시스템 역할을 설치합니다.

```
# yum install rhel-system-roles
```

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

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

5. 시나리오에 맞게 플레이북의 내용을 변경합니다. 예를 들어 다음 부분은 시스템이 **selinux-local-1.pp SELinux** 모듈을 설치하고 활성화하도록 합니다.

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

6. 변경 사항을 저장하고 텍스트 편집기를 종료합니다.

7. **host*1*, host*2* 및 *host3*** 시스템에서 플레이북을 실행합니다.

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

관련 자료

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

# 7장. 로깅 시스템 역할 사용

시스템 관리자는 로깅 시스템 역할을 사용하여 **RHEL** 호스트를 로깅 서버로 구성하여 여러 클라이언트 시스템에서 로그를 수집할 수 있습니다.

## 7.1. 로깅 시스템 역할

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

하나 이상의 시스템에 로깅 시스템 역할을 적용하려면 *플레이북에서* 로깅 구성을 정의합니다. 플레이북은 하나 이상의 플레이 목록입니다. 플레이북은 사람이 읽을 수 있으며 **YAML** 형식으로 작성됩니다. 플레이북에 대한 자세한 내용은 **Ansible** 설명서에서 플레이북 작업을 참조하십시오.

플레이북에 따라 **Ansible**에서 구성할 시스템 집합은 *인벤토리 파일에* 정의되어 있습니다. 인벤토리 생성 및 사용에 대한 자세한 내용은 **Ansible** 설명서에서 인벤토리를 빌드하는 방법을 참조하십시오.

로깅 솔루션은 여러 가지 방법으로 로그를 읽고 다양한 로깅 출력을 제공합니다.

예를 들어 로깅 시스템은 다음 입력을 수신할 수 있습니다.

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

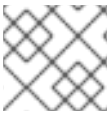
또한 로깅 시스템에는 다음과 같은 출력이 있을 수 있습니다.

- 로그는 **/var/log** 디렉토리의 로컬 파일에 저장됩니다.
- 로그는 **Elasticsearch**로 전송됩니다.
- 로그는 다른 로깅 시스템으로 전달됩니다.

로깅 시스템 역할을 사용하면 입력 및 출력을 필요에 맞게 결합할 수 있습니다. 예를 들어 로컬 파일의저널 입력을 저장하는 로깅 솔루션을 구성할 수 있지만 파일에서 읽은 입력은 모두 다른 로깅 시스템으로 전달되어 로컬 로그 파일에 저장됩니다.

## 7.2. 시스템 역할 매개변수 로깅

로깅 시스템 역할 플레이북에서는 **logging _inputs** 매개변수의 입력, **logging_outputs** 매개변수의 출력, **logging_flows** 매개변수의 입력과 출력 관계를 정의합니다.로깅 시스템 역할은 로깅 시스템을 구성하기 위해 추가 옵션을 사용하여 이러한 변수를 처리합니다. 암호화를 활성화할 수도 있습니다.



참고

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

- **logging_inputs**: 로깅 솔루션의 입력 목록.
  - **name**: 입력의 고유한 이름입니다.**logging_flows**: 입력 목록 및 생성된구성 파일 이름의 일부에 사용됩니다.
  - **type**: 입력 요소의 유형입니다. 유형은**roles/rsyslog/{tasks,vars}/inputs/** 의 디렉터리 이름에 해당하는 작업 유형을 지정합니다.

- 기본 사항: **systemd** 저널 또는 **unix** 소켓의 입력을 구성하는 입력.
  - **kernel_message**: **true**로 설정된 경우 **imklog** 를 로드합니다. 기본값은 **false**입니다.
  - **use_imuxsock**: **imjournal** 대신 **imuxsock** 을 사용합니다. 기본값은 **false**입니다.
  - **ratelimit_burst**: **ratelimit_interval** 내에서 내보낼 수 있는 최대 메시지 수. **use_imuxsock** 이 false인 경우 기본값은 **20000** 입니다. **use_imuxsock** 이 true인 경우 기본값은 **200** 입니다.
  - **ratelimit_interval**: **ratelimit_burst**를 평가하는 간격입니다. **use_imuxsock** 이 false인 경우 기본값은 600초입니다. **use_imuxsock** 이 true인 경우 기본값은 0입니다. 0은 속도 제한이 꺼져 있음을 나타냅니다.
  - **persist_state_interval**: 저널 상태는 모든 값 메시지에 유지됩니다. 기본값으로 **10** 입니다. **use_imuxsock** 이 false인 경우에만 유효합니다.
- 파일: 로컬 파일에서 입력을 구성하는 입력.
- **remote**: 네트워크를 통한 다른 로깅 시스템의 입력 구성 입력.

- **state**: 구성 파일의 상태입니다. **present** 또는 **absent** 입니다. 기본적으로 존재합니다.

- **logging_outputs**: 로깅 솔루션의 출력 목록입니다.
  - 파일: 로컬 파일에 출력을 구성하는 출력.
  - 전달: 출력을 다른 로깅 시스템으로 구성하는 출력.
  - **remote_files**: 다른 로깅 시스템에서 로컬 파일로 출력을 구성하는 출력.
- **logging_flows**: logging _**inputs**와 **logging_ outputs** 간의 관계를 정의하는 흐름 목록입니다. **logging_flows** 변수에는 다음 키가 있습니다.
  - **name**: 흐름의 고유한 이름
  - 입력: **logging_inputs** 이름 값 목록
  - 출력: **logging_outputs** 이름 값 목록입니다.

관련 자료

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

## 7.3. 로컬 로깅 시스템 역할 적용

다음 단계에 따라 **Red Hat Ansible Engine** 플레이북을 준비하고 적용하여 별도의 시스템 집합에서 로깅 솔루션을 구성합니다. 각 시스템은 로그를 로컬로 기록합니다.

사전 요구 사항

- **crypto_policies** 시스템 역할로 구성하려는 시스템인 하나 이상의*관리형 노드에* 대한 액세스 및 권한.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 *제어 노드에* 대한 액세스 및 권한. 제어 노드에서 다음을 수행합니다.

- Red Hat Ansible Engine 설치
- **rhel-system-roles** 패키지가 설치되어 있습니다.
- 관리 노드를 나열하는 인벤토리 파일.

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 내용을 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
```

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

검증

1. **/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. 시스템이 로그에 메시지를 전송하는지 확인합니다.
   a. 테스트 메시지를 전송합니다.

```
# 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 로거 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 **root** ).

## 7.4. 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 노드에* 대한 액세스 및 권한.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 *제어* 노드*에* 대한 액세스 및 권한. 제어 노드에서 다음을 수행합니다.
  - **Red Hat Ansible Engine** 설치
  - **rhel-system-roles** 패키지가 설치되어 있습니다.
  - 관리 노드를 나열하는 인벤토리 파일.

절차

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

```
---
- name: Deploying files input and configured files output
  hosts: all
  roles:
    - rhel-system-roles.logging
  vars:
    logging_inputs:
      - name: files_input0
        type: files
        input_log_path: /var/log/containerA/*.log
      - name: files_input1
        type: files
        input_log_path: /var/log/containerB/*.log
    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_input0, files_input1]
        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 필터링할 문자열로 바꿀 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/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. 시스템에서 오류 문자열이 포함된 메시지를 로그에 전송하는지 확인합니다.

   a. 테스트 메시지를 전송합니다.

```
# 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 로거 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 **root** ).

관련 자료

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

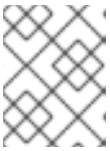
## 7.5. 로깅 시스템 역할을 사용하여 원격 로깅 솔루션 적용

다음 단계에 따라 **Red Hat Ansible Engine** 플레이북을 준비하고 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

디렉터리의 로컬 파일에 로그를 출력합니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.



참고

로깅 솔루션을 배포하려는 시스템에 **Red Hat Ansible Engine**을 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 있습니다.

참고

배포 시 시스템 역할이 **rsyslog** 를 설치하므로**rsyslog** 를 설치할 필요가 없습니다.

- 최소 두 개의 시스템이 있습니다.
  - 로깅 서버가 하나 이상 됩니다.
  - 로깅 클라이언트 중 하나는 하나 이상입니다.

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: servers
  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

요구 사항에 맞게 플레이북의 매개 변수를 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, 20514**가 포함됩니다. 다른 포트를 사용하려면 클라이언트 및 서버 시스템에서 **SELinux** 정책을 수정합니다. 시스템 역할을 통해 방화벽 구성는 아직 지원되지 않습니다.

2. 서버와 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 생성한 플레이북입니다.

검증

1. 클라이언트 및 서버 시스템 모두에서 **/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 전송합니다.

   ```
   # 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 로거 명령을 입력한 사용자의 사용자 이름이 포함됩니다(이 경우 **root** ).

관련 자료

- RHEL 시스템 역할 시작하기
- **/usr/share/ansible/roles/ rhel-system-roles.logging/README.md**의 **rhel-system-roles** 패키지와 함께 설치된 설명서
- RHEL 시스템 역할 KB 문서

## 7.6. 관련 자료

- RHEL 시스템 역할 시작하기.
- /usr/share/ansible/roles/ rhel-system-roles.logging/README.md의 rhel-system-roles 패키지와 함께 설치된 설명서.
- RHEL 시스템 역할 KB 문서.
- ansible-playbook(1) 도움말 페이지.

## 7.7. RELP에서 로깅 시스템 역할 사용

RELP(Reliable Event Logging Protocol)는 TCP 네트워크를 통한 데이터 및 메시지 로깅을 위한 네트워킹 프로토콜입니다. 이 솔루션은 이벤트 메시지를 안정적으로 전송하며 메시지 손실을 허용하지 않는 환경에서 사용할 수 있습니다.

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

RELP 클라이언트와 RELP 서버 간에 원격 로그인 시스템을 고려할 수 있습니다. RELP 클라이언트는 로그를 원격 로깅 시스템으로 전송하고 RELP 서버는 원격 로깅 시스템에서 보내는 모든 로그를 수신합니다.

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

### 7.7.1. RELP로 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 기록된 RHEL 시스템의 로그인을 구성하고 Ansible 플레이북을 실행하여 RELP를 사용하여 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 Ansible 인벤토리의 clients 그룹에 있는 모든 호스트에서 RELP를 구성합니다. RELP 구성은 TLS(전송 계층 보안)를 사용하여 네트워크를 통해 로그를 안전하게 전송할 수 있도록 메시지 전송을 암호화합니다.

사전 요구 사항

- RELP를 구성할 관리 노드에서 플레이북을 실행할 권한이 있습니다.
- 관리 노드는 제어 노드의 인벤토리 파일에 나열됩니다.
- ansible 및 rhel-system-roles 패키지는 제어 노드에 설치됩니다.

절차

1. 다음 내용으로 ***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]
```

Playbook은 다음 설정을 사용합니다.

- 대상: 이는 원격 로깅 시스템이 실행 중인 호스트 이름을 지정하는 필수 매개 변수입니다.
- 포트: 원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.
- **tls**: 네트워크를 통한 로그를 안전하게 전송합니다. 보안 래퍼를 사용하지 않으려면**tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로**tls** 매개변수는 **RELP**로 작업하는 동안 **true**로 설정되며 키/인증서 및 가입 항목 **{ca_cert,cert,private_key }** 및/또는 **{ca_cert_src,cert_src,private_key_src }**가 필요합니다.
  - **{ca_cert_src,cert_src,private_key_src}** Triplet가 설정된 경우 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 을 관리 노드의 대상으로 사용하여 제어 노드에서 파일을 전송합니다. 이 경우 파일 이름은 **Triplet**에서 원래 이름과 동일합니다.
  - **{ca_cert,cert,private_key}** Triplet가 설정된 경우 로깅 구성 전에 파일이 기본 경로에 있어야 합니다.
  - 두 트라이블릿이 모두 설정되면 파일이 제어 노드에서 관리 노드의 특정 경로로 로컬 경로로 전송됩니다.
- **ca_cert**: CA 인증서 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.
- 인증서: 인증서 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.
- **private_key**: 개인 키의 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.
- **ca_cert_src**: Repcurrents 로컬 CA 인증서 파일 경로는 대상 호스트에 복사됩니다. **ca_cert**를 지정하면 위치에 복사됩니다.
- **cert_src**: 대상 호스트에 복사되는 로컬 인증서 파일 경로를 복제합니다. 인증서가 지정되면 위치에 복사됩니다.
- **private_key_src**: 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key**가 지정되면 위치에 복사됩니다.
- **pki_authmode**: 은 인증 모드를이름 또는지문 으로 허용합니다.

- **permitted_servers**: 로깅 클라이언트에서 **TLS**를 통해 연결하고 로그를 전송하도록 허용하는 서버 목록입니다.
- 입력: 로깅 입력 사전 목록.
- 출력: 로깅 출력 사전 목록.

2. 선택 사항: 플레이북 구문을 확인합니다.

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

3. 플레이북을 실행합니다.

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

### 7.7.2. RELP로 서버 로깅 구성

로깅 시스템 역할을 사용하여 **RHEL** 시스템의 로그인을 서버로 구성하고 **Ansible** 플레이북을 실행하여 **RELP**로 원격 로깅 시스템에서 로그를 수신할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 서버 그룹에 있는 모든 호스트에서 **RELP**를 구성합니다. **RELP** 구성은 **TLS**를 사용하여 네트워크를 통해 로그를 안전하게 전송할 수 있도록 메시지 전송을 암호화합니다.

사전 요구 사항

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

절차

1. 다음 내용으로 ***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_
```

Playbook은 다음 설정을 사용합니다.

- 포트: 원격 로깅 시스템이 수신 대기 중인 포트 번호입니다.
- **tls**: 네트워크를 통한 로그를 안전하게 전송합니다. 보안 래퍼를 사용하지 않으려면**tls** 변수를 **false** 로 설정할 수 있습니다. 기본적으로**tls** 매개변수는 **RELP**로 작업하는 동안 **true**로 설정되며 키/인증서 및 가입 항목 **{ca_cert,cert,private_key }** 및/또는 **{ca_cert_src,cert_src,private_key_src }**가 필요합니다.
  - **{ca_cert_src,cert_src,private_key_src}** Triplet가 설정된 경우 기본 위치 **/etc/pki/tls/certs** 및 **/etc/pki/tls/private** 을 관리 노드의 대상으로 사용하여 제어 노드에서 파일을 전송합니다. 이 경우 파일 이름은 **Triplet**에서 원래 이름과 동일합니다.
  - **{ca_cert,cert,private_key}** Triplet가 설정된 경우 로깅 구성 전에 파일이 기본 경로에 있어야 합니다.
  - 두 트라이블릿이 모두 설정되면 파일이 제어 노드에서 관리 노드의 특정 경로로 로컬 경로로 전송됩니다.
- **ca_cert**: CA 인증서 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/certs/ca.pem** 이며 파일 이름은 사용자가 설정합니다.
- 인증서: 인증서 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/certs/server-cert.pem** 이며 파일 이름은 사용자가 설정합니다.
- **private_key**: 개인 키의 경로를 나타냅니다. 기본 경로는**/etc/pki/tls/private/server-key.pem** 이며 파일 이름은 사용자가 설정합니다.
- **ca_cert_src**: Repcurrents 로컬 CA 인증서 파일 경로는 대상 호스트에 복사됩니다. **ca_cert**를 지정하면 위치에 복사됩니다.
- **cert_src**: 대상 호스트에 복사되는 로컬 인증서 파일 경로를 복제합니다. 인증서가 지정되면 위치에 복사됩니다.
- **private_key_src**: 대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key**가 지정되면 위치에 복사됩니다.
- **pki_authmode**: 은 인증 모드를이름 또는지문 으로 허용합니다.
- **permitted_clients**: 로깅 서버가 **TLS**를 통해 연결하고 로그를 전송하도록 허용하는 클라이언트 목록입니다.
- 입력: 로깅 입력 사전 목록.
- 출력: 로깅 출력 사전 목록.

2. 선택 사항: 플레이북 구문을 확인합니다.

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

3. 플레이북을 실행합니다.

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

## 7.8. TLS에서 로깅 시스템 역할 사용

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

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

### 7.8.1. TLS를 사용하여 클라이언트 로깅 구성

로깅 시스템 역할을 사용하여 로컬 시스템에 기록된 **RHEL** 시스템의 로그인을 구성하고 **Ansible** 플레이북을 실행하여 **TLS**를 사용하여 원격 로깅 시스템으로 로그를 전송할 수 있습니다.

이 절차에서는 **Ansible** 인벤토리의 **clients** 그룹에 있는 모든 호스트에서 **TLS**를 구성합니다. **TLS** 프로토콜은 네트워크를 통해 로그를 안전하게 전송할 수 있도록 메시지 전송을 암호화합니다.

사전 요구 사항

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

절차

1. 다음 내용으로 ***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]
```

Playbook은 다음 매개 변수를 사용합니다.

**logging_pki_files**

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

**ca_cert**

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

인증서

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

**private_key**

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

**ca_cert_src**

Repcurrents 로컬 CA 인증서 파일 경로는 대상 호스트에 복사됩니다. **ca_cert** 를 지정하면 위치에 복사됩니다.

**cert_src**

대상 호스트에 복사되는 로컬 인증서 파일 경로를 복제합니다. 인증서가 지정되면 위치에 복사됩니다.

**private_key_src**

대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 가 지정되면 위치에 복사됩니다.

**tls**

이 매개 변수를 사용하면 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 사용하지 않으려면 **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
```

### 7.8.2. TLS를 사용하여 서버 로깅 구성

로깅 시스템 역할을 사용하여 RHEL 시스템의 로그인을 서버로 구성하고, Ansible 플레이북을 실행하여 TLS를 사용하여 원격 로깅 시스템에서 로그를 수신할 수 있습니다.

이 절차에서는 Ansible 인벤토리의 서버 그룹에 있는 모든 호스트에서 TLS를 구성합니다.

사전 요구 사항

- TLS를 구성할 관리 노드에서 플레이북을 실행할 권한이 있습니다.

- 관리 노드는 제어 노드의 인벤토리 파일에 나열됩니다.
- **ansible** 및 **rhel-system-roles** 패키지는 제어 노드에 설치됩니다.

절차

1. 다음 내용으로 ***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]
```

Playbook은 다음 매개 변수를 사용합니다.

**logging_pki_files**

이 매개변수를 사용하여 **TLS**를 구성할 수 있으며 **ca_cert_src,cert_src** 및 **private_key_src** 매개변수를 전달해야 합니다.

**ca_cert**

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

인증서

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

**private_key**

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

**ca_cert_src**

Repcurrents 로컬 CA 인증서 파일 경로는 대상 호스트에 복사됩니다. **ca_cert** 를 지정하면 위치에 복사됩니다.

**cert_src**

대상 호스트에 복사되는 로컬 인증서 파일 경로를 복제합니다. 인증서가 지정되면 위치에 복사됩니다.

**private_key_src**

대상 호스트에 복사되는 로컬 키 파일 경로를 나타냅니다. **private_key** 가 지정되면 위치에 복사됩니다.

**tls**

이 매개 변수를 사용하면 네트워크를 통해 로그를 안전하게 전송할 수 있습니다. 보안 래퍼를 사용하지 않으려면 **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
```

# 8장. SSH 시스템 역할을 사용하여 보안 통신 구성

관리자는 **sshd** 시스템 역할을 사용하여 **SSH** 서버를 구성하고**ssh** 시스템 역할을 사용하여 **Red Hat Ansible Automation Platform**을 사용하여 동시에 여러 **RHEL** 시스템에서 **SSH** 클라이언트를 일관되게 설정할 수 있습니다.

## 8.1. SSHD 시스템 역할 변수

**sshd** 시스템 역할 플레이북에서는 기본 설정 및 제한 사항에 따라 **SSH** 구성 파일에 대한 매개변수를 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 들어 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**

**False** 로 설정하면 역할이 완전히 비활성화됩니다. 기본값은**True** 입니다.

**sshd_skip_defaults**

**True** 로 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**

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

**sshd_allow_reload**

**False** 로 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**

**True** 로 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 들어 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 들어 다음과 같습니다.

```
sshd_Compression: no
```

**sshd_match** and **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.

**sshd** 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_binary**

**openssh** 의 **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** 서비스의 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**

RHEL 8에서 **true** 로 설정하면 이 변수가 시스템 전체의 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.

## 8.2. SSH 서버 시스템 역할을 사용하여 OPENSSH 서버 구성

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

사전 요구 사항

- SSH 서버 시스템 역할을 사용하여 구성하려는 시스템인 하나 이상의 *관리형 노드에* 대한 액세스 및 권한.
- Red Hat Ansible Engine이 다른 시스템을 구성하는 시스템인 *제어 노드에* 대한 액세스 및 권한. 제어 노드에서 다음을 수행합니다.
  - Red Hat Ansible Engine이 설치되어 있어야 합니다.
  - **rhel-system-roles** 패키지가 설치되어 있습니다.
  - 관리 노드를 나열하는 인벤토리 파일.

절차

1. SSH 서버 시스템 역할에 대한 예제 플레이북을 복사합니다.

```
# cp /usr/share/doc/rhel-system-roles-VERSION/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
```

Playbook은 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** 서버 시스템 역할 변수를 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 **root**로 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. **root** 로 서버에 연결합니다:

   ```
   $ ssh root@10.1.1.1
   ```

관련 자료

- **/usr/share/doc/rhel-system-roles-VERSION/sshd/README.md** file.
- **ansible-playbook(1)** 도움말 페이지.

## 8.3. SSH 클라이언트 시스템 역할 변수

**SSH** 시스템 역할 플레이북에서는 환경 설정 및 제한 사항에 따라 클라이언트 **SSH** 구성 파일에 대한 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 들어 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. **simple** 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.

## 8.4. SSH 시스템 역할을 사용하여 OPENSSH 클라이언트 구성

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

사전 요구 사항

- **ssh** 시스템 역할로 구성하려는 시스템인 하나 이상의*관리형 노드에* 대한 액세스 및 권한.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 시스템인 *제어 노드에* 대한 액세스 및 권한. 제어 노드에서 다음을 수행합니다.
  - **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
  - **rhel-system-roles** 패키지가 설치되어 있습니다.
  - 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_ForwardX11: 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 별칭이 생성되어* ***example.com*** 호스트와 ***user1*** *사용자* 이름이 인 에 대한 연결을 나타냅니다.
- **X11** 전달이 비활성화되어 있습니다.

선택적으로 환경 설정에 따라 이러한 변수를 수정할 수 있습니다. 자세한 내용은 **ssh** 역할 변수를 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
```

# 9장. 시스템 전체에 사용자 정의 암호화 정책 설정

관리자는 RHEL에서 **crypto_policies** 시스템 역할을 사용하여 Red Hat Ansible Automation Platform을 사용하여 다양한 시스템에서 사용자 정의 암호화 정책을 신속하고 일관되게 구성할 수 있습니다.

## 9.1. 암호화 정책 시스템 역할 변수 및 정보

Policies System Role Playbook에서는 환경 설정 및 제한 사항에 따라 암호화 정책 구성 파일의 매개 변수를 정의할 수 있습니다.

변수를 구성하지 않으면 시스템 역할은 시스템을 구성하지 않고 팩트만 보고합니다.

정책 시스템 역할에 대해 선택한 변수

**crypto_policies_policy**

시스템 역할이 관리되는 노드에 적용되는 암호화 정책 수준을 결정합니다. 다양한 암호화 정책 수준에 대한 자세한 내용은 시스템 전체 암호화 정책을 참조하십시오.

**crypto_policies_reload**

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

**crypto_policies_reboot_ok**

**yes** 로 설정하고 시스템 역할이 **crypto** 정책을 변경한 후 재부팅해야 하는 경우 **crypto_policies_reboot_required** 를 **yes** 로 설정합니다. 기본값은 **no** 입니다.

정책 시스템 역할에 의해 설정된 팩트

**crypto_policies_active**

현재 선택한 정책을 나열합니다.

**crypto_policies_available_policies**

시스템에서 사용 가능한 모든 정책 수준을 나열합니다.

**crypto_policies_available_modules**

시스템에서 사용 가능한 모든 하위 정책 모듈을 나열합니다.

관련 자료

* 사용자 정의 시스템 전체 암호화 정책 생성 및 설정

## 9.2. POLICIES SYSTEM ROLE을 사용하여 사용자 정의 암호화 정책 설정

**crypto_policies** 시스템 역할을 사용하여 단일 제어 노드에서 일관되게 많은 관리되는 노드를 구성할 수 있습니다.

사전 요구 사항

- Policies 시스템 역할로 구성하려는 시스템인 하나 이상의 *관리형 노드에* 대한 액세스 및 권한.
- Red Hat Ansible Engine이 다른 시스템을 구성하는 시스템인 *제어 노드에* 대한 액세스 및 권한.
  제어 노드에서 다음을 수행합니다.
  - Red Hat Ansible Engine 설치

- **rhel-system-roles** 패키지가 설치되어 있습니다.
- 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 선호하는 암호화 정책으로 교체할 수 있습니다.**DEFAULT,LEGACY** 및 **FIPS:OSPP**.

**crypto_policies_reboot_ok: true** 변수를 사용하면 시스템 역할이 **crypto** 정책을 변경한 후 시스템이 재부팅됩니다.

자세한 내용은 Policies Policies System Role variables and facts를 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 인 다른 플레이북(예: ***verify_playbook.yml)을 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 않고 관리 노드의 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 활성으로 표시합니다.

## 9.3. 관련 자료

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

# 10장. NBDE_CLIENT 및 NBDE_SERVER 시스템 역할 사용

## 10.1. NBDE_CLIENT 및 NBDE_SERVER 시스템 역할 소개

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

RHEL 7.9에서는 Clevis 및 Tang을 사용하여 PBD(Policy-Based Decryption) 솔루션의 자동화된 배포를 위한 Ansible 역할을 도입했습니다. **rhel-system-roles** 패키지에는 이러한 시스템 역할, 관련 예제 및 참조 문서가 포함되어 있습니다.

**nbde_client** 시스템 역할을 사용하면 자동화된 방식으로 여러 Clevis 클라이언트를 배포할 수 있습니다. **nbde_client** 역할은 Tang 바인딩만 지원하며 현재 TPM2 바인딩에는 사용할 수 없습니다.

**nbde_client** 역할에는 LUKS를 사용하여 이미 암호화된 볼륨이 필요합니다. 이 역할은 LUKS 암호화된 볼륨을 하나 이상의 NBDE(Network-Bound) 서버 - Tang 서버에 바인딩하도록 지원합니다. 기존 볼륨 암호화를 암호로 보존하거나 제거할 수 있습니다. 암호를 제거한 후 NBDE만 사용하여 볼륨을 잠금 해제할 수 있습니다. 이 기능은 시스템을 프로비저닝한 후 제거해야 하는 임시 키 또는 암호를 사용하여 볼륨을 처음 암호화할 때 유용합니다.

암호와 키 파일을 둘 다 제공하면 이 역할은 먼저 제공한 정보를 사용합니다. 유효한 이러한 항목을 찾지 못하면 기존 바인딩에서 암호를 검색하려고 합니다.

PBD는 바인딩을 슬롯에 대한 장치의 매핑으로 정의합니다. 즉, 동일한 장치에 대해 여러 바인딩을 가질 수 있습니다. 기본 슬롯은 슬롯 1입니다.

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

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

- Tang 키 순환
- Tang 키 배포 및 백업

관련 자료

- NBDE(Network-Bound Disk Encryption) 역할 변수에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles-VERSION/nbde_client/** 및 **/usr/share/rhel-system-roles-VERSION/nbde_client/** 의 **README.md** 파일을 참조하십시오.
- 예를 들어 system-roles 플레이북을 설치하고 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles-VERSION/ndbe_server/example-*-playbook.yml** 및 **/usr/share/doc/rhel-system-roles-VERSION/nbde_client/example-*-playbook.yml** 디렉터리를 확인합니다.
- RHEL 시스템 역할에 대한 자세한 내용은 RHEL 시스템 역할 소개를참조하십시오.

## 10.2. NBDE_SERVER 시스템 역할을 사용하여 여러 TANG 서버 설정

단계를 수행하여 Tang-server 설정이 포함된 Ansible 플레이북을 준비하고 적용합니다.

사전 요구 사항

- **Red Hat Ansible Engine** 서브스크립션이 시스템에 연결되어 있습니다. 자세한 내용은 How do I download and install Red Hat Ansible Engine 문서를 참조하십시오.

절차

1. **RHEL Ansible** 리포지토리를 활성화합니다. 예를 들면 다음과 같습니다.

```
# sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

2. **Ansible Engine**을 설치합니다.

```
# yum install ansible
```

3. **RHEL** 시스템 역할을 설치합니다.

```
# yum install rhel-system-roles
```

4. **Tang** 서버에 대한 설정이 포함된 플레이북을 준비합니다. 처음부터 시작하거나 **/usr/share/doc/rhel-system-roles-VERSION/nbde_server/** 디렉터리에서 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/doc/rhel-system-roles-VERSION/nbde_server/example-simple_deploy-playbook.yml ./my-tang-playbook.yml
```

5. 선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

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

6. 필수 매개 변수를 추가합니다. 다음 예제 플레이북에서는 **Tang** 서버 및 키 순환을 배포합니다.

```
---
- hosts: all

  vars:
    nbde_server_rotate_keys: yes

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

7. 완료된 플레이북을 적용합니다.

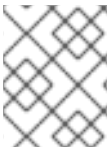
```
# ansible-playbook -i host1,host2,host3 my-tang-playbook.yml
```

관련 자료

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

## 10.3. NBDE_CLIENT 시스템 역할을 사용하여 여러 CLEVIS 클라이언트 설정

단계를 수행하여 **Clevis-client** 설정이 포함된 **Ansible** 플레이북을 준비하고 적용합니다.



참고

**nbde_client** 시스템 역할은 **Tang** 바인딩만 지원합니다. 즉, 현재 **TPM2** 바인딩에 사용할 수 없습니다.

사전 요구 사항

- **Red Hat Ansible Engine** 서브스크립션이 시스템에 연결되어 있습니다. 자세한 내용은 How do I download and install Red Hat Ansible Engine 문서를 참조하십시오.
- 볼륨은 **LUKS**에서 이미 암호화되어 있습니다.

절차

1. **RHEL Ansible** 리포지토리를 활성화합니다. 예를 들면 다음과 같습니다.

```
# sudo subscription-manager repos --enable rhel-7-server-ansible-2.9-rpms
```

2. **Ansible Engine**을 설치합니다.

```
# yum install ansible
```

3. **RHEL** 시스템 역할을 설치합니다.

```
# yum install rhel-system-roles
```

4. **Clevis** 클라이언트에 대한 설정이 포함된 플레이북을 준비합니다. 처음부터 시작하거나 **/usr/share/doc/rhel-system-roles-VERSION/nbde_client/** 디렉터리 디렉터리에서 예제 플레이북 중 하나를 사용할 수 있습니다.

```
# cp /usr/share/doc/rhel-system-roles-VERSION/nbde_client/example-high_availability-playbook.yml ./my-clevis-playbook.yml
```

5. 선택한 텍스트 편집기에서 플레이북을 편집합니다. 예를 들면 다음과 같습니다.

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

6. 필수 매개 변수를 추가합니다. 다음 예제 플레이북에서는 두 개의 **Tang** 서버 중 하나를 사용할 수 있는 경우 두 개의 **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
```

7. 완료된 플레이북을 적용합니다.

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

관련 자료

- Clevis 클라이언트 시스템 역할에 대한 매개변수 및 추가 정보에 대한 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles-VERSION/nbde_client/** 디렉터리를 참조하십시오.

# 11장. RHEL 시스템 역할을 사용하여 인증서 요청

인증서 시스템 역할을 사용하면 **Red Hat Ansible Engine**을 사용하여 인증서를 발행하고 관리할 수 있습니다.

이 장에서는 다음 주제를 다룹니다.

- 인증서 시스템 역할
- 인증서 시스템 역할을 사용하여 새 자체 서명인증서 요청
- 인증서 시스템 역할을 사용하여 **IdM CA**에서 새인증서 요청

## 11.1. 인증서 시스템 역할

인증서 시스템 역할을 사용하면 **Red Hat Ansible Engine**을 사용하여 **TLS** 및 **SSL** 인증서를 발행 및 갱신할 수 있습니다.

이 역할은 인증서 프로바이더로 **certmonger** 를 사용하며 현재 자체 서명된 인증서 및 **IdM** 통합 **CA(**인증 기관**)**를 사용하여 갱신을 지원합니다.

**Certificate System** 역할과 함께 **Ansible** 플레이북에서 다음 변수를 사용할 수 있습니다.

- **certificate_wait** 를 통해 작업이 인증서가 발행될 때까지 기다려야 하는지를 지정합니다.
- 실행할 각 인증서와 해당 매개 변수를 나타내는 **certificate_requests**입니다.

관련 자료

- **certificate _requests** 변수에 사용되는 매개변수 및 인증서 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 파일을 참조하십시오.
- **RHEL** 시스템 역할 및 해당 역할을 적용하는 방법에 대한 자세한 내용은 **RHEL** 시스템 역할 시작하기 를 참조하십시오.

## 11.2. 인증서 시스템 역할을 사용하여 새 자체 서명인증서 요청

인증서 시스템 역할을 사용하면 **Red Hat Ansible Engine**을 사용하여 자체 서명된 인증서를 발행할 수 있습니다.

이 프로세스는 **certmonger** 프로바이더를 사용하고**getcert** 명령을 통해 인증서를 요청합니다.

참고

기본적으로 **certmonger** 는 만료되기 전에 자동으로 인증서를 갱신하려고 합니다. **Ansible** 플레이북에서 **auto_renew** 매개 변수를**no** 로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.

참고

인증서 솔루션을 배포하려는 시스템에 **Ansible**을 설치할 필요는 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.

절차

1. *선택 사항:* 인벤토리 파일을 생성합니다(예:**inventory.file**:

```
$ touch inventory.file
```

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

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

3. 플레이북 파일을 생성합니다(예: **request-certificate.yml** ):
   - 인증서를 요청하려는 호스트를 포함 하도록 호스트를 설정합니다(예:**webserver** ).
   - 다음을 포함하도록 **certificate_requests** 변수를 설정합니다.
     - **name** 매개 변수를**mycert** 와 같이 원하는 인증서 이름으로 설정합니다.
     - **dns** 매개 변수를 인증서에 포함할 도메인(예:***.example.com** )으로 설정합니다.
     - **ca** 매개 변수를**self-sign** 으로 설정합니다.
   - 역할에서 **rhel-system-roles.certificate** 역할을 설정합니다.
     다음은 이 예제의 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
```

관련 자료

- **certificate _requests** 변수에 사용되는 매개변수 및 인증서 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 파일을 참조하십시오.

- **ansible-playbook** 명령에 대한 자세한 내용은**ansible-playbook(1)** 도움말 페이지를 참조하십시오.

## 11.3. 인증서 시스템 역할을 사용하여 IDM CA에서 새 인증서 요청

인증 시스템 역할을 사용하면 **Red Hat Ansible Engine**을 사용하여 통합 인증 기관**(CA)**이 있는 **IdM** 서버를 사용하는 동안 인증서를 발행할 수 있습니다. 따라서 **IdM**을 **CA**로 사용할 때 여러 시스템의 인증서 신뢰 체인을 효율적이고 일관되게 관리할 수 있습니다.

이 프로세스는 **certmonger** 프로바이더를 사용하고**getcert** 명령을 통해 인증서를 요청합니다.

참고

기본적으로 **certmonger** 는 만료되기 전에 자동으로 인증서를 갱신하려고 합니다. **Ansible** 플레이북에서 **auto_renew** 매개 변수를**no** 로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.

  참고

  인증서 솔루션을 배포하려는 시스템에 **Ansible**을 설치할 필요는 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.

절차

1. *선택 사항:* 인벤토리 파일을 생성합니다(예:**inventory.file**:

   ```
   $ touch inventory.file
   ```

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

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

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

   - 인증서를 요청하려는 호스트를 포함 하도록 호스트를 설정합니다(예:**webserver** ).
   - 다음을 포함하도록 **certificate_requests** 변수를 설정합니다.
     - **name** 매개 변수를**mycert** 와 같이 원하는 인증서 이름으로 설정합니다.
     - **dns** 매개 변수를 인증서에 포함할 도메인(예:**www.example.com)** 으로 설정합니다.
     - **principal** 매개 변수를 설정하여 Kerberos 주체(예: **HTTP/www.example.com@EXAMPLE.COM)**를 지정합니다.
     - **ca** 매개 변수를**ipa** 로 설정합니다.

- 역할에서 **rhel-system-roles.certificate** 역할을 설정합니다.
  다음은 이 예제의 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
```

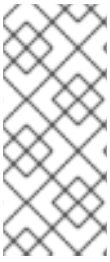
관련 자료

- **certificate _requests** 변수에 사용되는 매개변수 및 인증서 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 파일을 참조하십시오.
- **ansible-playbook** 명령에 대한 자세한 내용은**ansible-playbook(1)** 도움말 페이지를 참조하십시오.

## 11.4. 인증서 시스템 역할을 사용하여 인증서 발급 전 또는 이후에 실행할 명령 지정

인증서 시스템 역할을 사용하면 **Red Hat Ansible Engine**을 사용하여 인증서를 발행하거나 갱신하기 전과 후에 명령을 실행할 수 있습니다.

다음 예에서 관리자는 **www.example.com** 의 자체 서명된 인증서가 발급되거나 갱신되기 전에 **httpd** 서비스를 중지하고 나중에 다시 시작합니다.



참고

기본적으로 **certmonger** 는 만료되기 전에 자동으로 인증서를 갱신하려고 합니다. **Ansible** 플레이북에서 **auto_renew** 매개 변수를 **no** 로 설정하여 이 설정을 비활성화할 수 있습니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.

참고

인증서 솔루션을 배포하려는 시스템에 **Ansible**을 설치할 필요는 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.

절차

1. *선택 사항:* 인벤토리 파일을 생성합니다(예: **inventory.file**:

```
$ touch inventory.file
```

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

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

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

- 인증서를 요청하려는 호스트를 포함 하도록 호스트를 설정합니다(예: **webserver** ).
- 다음을 포함하도록 **certificate_requests** 변수를 설정합니다.
  - **name** 매개 변수를 **mycert** 와 같이 원하는 인증서 이름으로 설정합니다.
  - **dns** 매개 변수를 인증서에 포함할 도메인(예: **www.example.com**) 으로 설정합니다.

- 인증서를 발급하는 데 사용할 **ca** 매개 변수(예: 자체 서명 )를 설정합니다.
- **run_before** 매개변수를 **systemctl stop httpd.service** 와 같이 인증서가 발행되거나 갱신되기 전에 실행할 명령으로 설정합니다.
- 이 인증서가 발급되거나 갱신된 후 **systemctl start httpd.service** 와 같이 **run_after** 매개 변수를 실행할 명령으로 설정합니다.

- 역할에서 **rhel-system-roles.certificate** 역할을 설정합니다 .

  다음은 이 예제의 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
```

관련 자료

- **certificate _requests** 변수에 사용되는 매개변수 및 인증서 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.certificate/README.md** 파일을 참조하십시오.
- **ansible-playbook** 명령에 대한 자세한 내용은 **ansible-playbook(1)** 도움말 페이지를 참조하십시오.

# 12장. RHEL 시스템 역할을 사용하여 KDUMP 구성

**Ansible**을 사용하여 **kdump**를 관리하려면 **RHEL 7.9**에서 사용할 수 있는 **RHEL** 시스템 역할 중 하나인 **Kdump** 역할을 사용할 수 있습니다.

**kdump** 를 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 소개를 참조하십시오.

## 12.1. KDUMP RHEL 시스템 역할

**Kdump** 시스템 역할을 사용하면 여러 시스템에서 기본 커널 덤프 매개 변수를 설정할 수 있습니다.

## 12.2. KDUMP 역할 매개변수

**Kdump RHEL** 시스템 역할에 사용되는 매개변수는 다음과 같습니다.

| 역할 변수 | Description |
|---|---|
| kdump_path | **vmcore** 가 작성된 경로입니다.**kdump_target** 이 null이 아닌 경우 경로는 해당 덤프 대상을 기준으로 합니다. 그렇지 않으면 루트 파일 시스템의 절대 경로여야 합니다. |

관련 자료

- **makedumpfile(8)** 도움말 페이지를 참조하십시오.
- **kdump** 에 사용되는 매개변수 및 **Kdump** 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.kdump/README.md** 파일을 참조하십시오.

## 12.3. RHEL 시스템 역할을 사용하여 KDUMP 구성

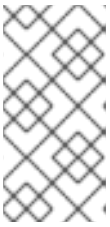
**Ansible** 플레이북을 실행하여 **Kdump** 시스템 역할을 사용하여 여러 시스템에서 기본 커널 덤프 매개 변수를 설정할 수 있습니다.

주의

**Kdump** 역할은 **/etc/kdump.conf** 파일을 교체하여 전체적으로 관리 호스트의 **kdump** 설정을 대체합니다. 또한 **Kdump** 역할이 적용된 경우 **/etc/sysconfig/kdump** 파일을 교체하여 역할 변수에서 지정하지 않은 경우에도 이전 **kdump** 설정도 모두 교체됩니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.



참고

**kdump** 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다.

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **kdump** 를 배포할 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
```

관련 자료

- **kdump** 역할 변수에 대한 자세한 내용은 **/usr/share/doc/rhel-system-roles-VERSION/kdump** 디렉터리의 **README.md** 파일을 참조하십시오.
- 역할 적용을 참조하십시오.
- **rhel-system-roles** 패키지 **/usr/share/ansible/roles/rhel-system-roles.kdump/README.md**와 함께 설치된 설명서

# 13장. RHEL 시스템 역할을 사용하여 로컬 스토리지 관리

**Ansible**을 사용하여 **LVM** 및 로컬 파일 시스템(**FS**)을 관리하려면 **RHEL 7.9**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 소개를 참조하십시오.

## 13.1. 스토리지 역할 소개

스토리지 역할은 다음을 관리할 수 있습니다.

- 파티션되지 않은 디스크의 파일 시스템
- 논리 볼륨 및 파일 시스템을 포함한 전체 **LVM** 볼륨 그룹

**Storage** 역할을 사용하면 다음 작업을 수행할 수 있습니다.

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

- **LVM** 볼륨 그룹 제거
- 논리 볼륨 만들기
- 논리 볼륨 제거
- **RAID** 볼륨 만들기
- **RAID** 볼륨 제거
- **RAID**를 사용하여 **LVM** 풀 만들기
- **RAID**를 사용하여 **LVM** 풀 제거

## 13.2. 블록 장치에서 XFS 파일 시스템을 생성하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 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 **13.1. /dev/sdb**에 **XFS**를 생성하는 플레이북

```
---
- hosts: all
  vars:
    storage_volumes:
      - name: barefs
        type: disk
```

```
    disks:
      - sdb
    fs_type: xfs
roles:
  - rhel-system-roles.storage
```

- 볼륨 이름(예의 ***barefs***)은 현재 임의입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.
- **XFS**는 **RHEL 7.9**의 기본 파일 시스템이므로 **fs_type: xfs** 행을 생략할 수 있습니다.
- **LV**에서 파일 시스템을 생성하려면 포함 볼륨 그룹을 포함하여 **disks:** 속성 아래에 **LVM** 설정을 제공합니다. 자세한 내용은 논리 볼륨을 관리하기 위한 **Ansible** 플레이북 예제를 참조하십시오.

  **LV** 장치의 경로를 제공하지 마십시오.

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.3. 파일 시스템을 영구적으로 마운트하는 ANSIBLE 플레이북의 예

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

예 **13.2. /dev/sdb**에 파일 시스템을 마운트하는 플레이북을 **/mnt/data**에 마운트합니다.

```
---
- 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** 장치의 파일 시스템 또는 마운트 지점 디렉터리가 없으면 플레이북에서 생성합니다.

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.4. 논리 볼륨을 관리하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다. 이 플레이북은 스토리지 역할을 적용하여 볼륨 그룹에 **LVM** 논리 볼륨을 만듭니다.

예 **13.3. 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 이를 생성합니다.

- 플레이북은 **mylv** 논리 볼륨에 **Ext4** 파일 시스템을 생성하고 **/mnt** 에 파일 시스템을 영구적으로 마운트합니다.

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.5. 온라인 블록 삭제를 활성화하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다. 이 플레이북은 온라인 블록 삭제가 활성화된 **XFS** 파일 시스템을 마운트하는 **Storage** 역할을 적용합니다.

예 **13.4. /mnt/data/**에서 온라인 블록 삭제를 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
```

관련 자료

- 또한 이 플레이북은 **Example Ansible** 플레이북에 설명된 영구 마운트 예제의 모든 작업을 수행하여 파일 시스템을 지속적으로 마운트합니다.

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오**.**

### 13.6. EXT4 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다**.** 이 플레이북은 **Ext4** 파일 시스템을 만들고 마운트하기 위해 **Storage** 역할을 적용합니다**.**

예 **13.5. /dev/sdb**에서 **Ext4**를 생성하고 **/mnt/data**에 마운트하는 플레이북

```
---
- 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**.**

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오**.**

### 13.7. EXT3 파일 시스템을 생성하고 마운트하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다**.** 이 플레이북은 **Ext3** 파일 시스템을 만들고 마운

트하기 위해 **Storage** 역할을 적용합니다.

예 **13.6. /dev/sdb**에서 **Ext3**을 생성하고 **/mnt/data**에 마운트하는 플레이북

```
---
- 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.

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.8. 스토리지 RHEL 시스템 역할을 사용하여 기존 EXT4 또는 EXT3 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북의 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다. 이 **Playbook**은 블록 장치의 기존 **Ext4** 또는 **Ext3** 파일 시스템의 크기를 조정하는 **Storage** 역할을 적용합니다.

예 **13.7.** 디스크에 단일 볼륨을 설정하는 플레이북

```
---
- name: Create a disk device mounted on /opt/barefs
- hosts: all
  vars:
    storage_volumes:
```

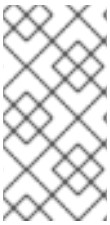
```
    - name: barefs
      type: disk
      disks:
        - /dev/sdb
  size: 12 GiB
      fs_type: ext4
      mount_point: /opt/barefs
  roles:
    - rhel-system-roles.storage
```

- 이전 예제의 볼륨이 이미 있는 경우 볼륨 크기를 조정하려면 매개 변수 크기에 대해 다른 값을 사용하여 동일한 플레이북을 실행해야 합니다. 예를 들어 다음과 같습니다.

예 **13.8. /dev/sdb**에서 **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(예의 모음)은 현재 임의입니다. **Storage** 역할은 **disks:** 속성 아래에 나열된 디스크 장치에서 볼륨을 식별합니다.



참고

다른 파일 시스템에서 크기 조정 작업을 사용하면 작업 중인 장치의 데이터를 삭제할 수 있습니다.

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.9. 스토리지 RHEL 시스템 역할을 사용하여 LVM의 기존 파일 시스템의 크기를 조정하는 ANSIBLE 플레이북 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다. 이 플레이북은 스토리지 **RHEL** 시스템 역할을 적용하여 파일 시스템으로 **LVM** 논리 볼륨의 크기를 조정합니다.

주의

다른 파일 시스템에서 크기 조정 작업을 사용하면 작업 중인 장치의 데이터를 삭제할 수 있습니다.

예 **13.9. myvg** 볼륨 그룹에서 기존 **mylv1** 및 **myvl2** 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
  incude_role:
    name: rhel-system-roles.storage
```

- 이 플레이북은 다음과 같은 기존 파일 시스템의 크기를 조정합니다.
  - **/opt/mount 1**에 마운트된 **mylv1** 볼륨의 **Ext4** 파일 시스템은 **10GiB**로 조정합니다.

- **/opt/mount 2**에 마운트된 **mylv2** 볼륨의 **Ext4** 파일 시스템은 **50GiB**로 조정됩니다**.**

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오**.**

## 13.10. 스토리지 RHEL 시스템 역할을 사용하여 스왑 파티션을 생성하는 ANSIBLE 플레이북 예

이 섹션에서는 **Ansible** 플레이북의 예를 제공합니다**.** 이 **Playbook**은 **Storage** 역할을 적용하여 스왑 파티션을 만들거나 기본 매개 변수를 사용하여 블록 장치에 이미 존재하는 경우 스왑 파티션을 수정합니다**.**

예 **13.10. /dev/sdb**에서 기존 **XFS**를 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**.**

관련 자료

- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오**.**

## 13.11. 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 사용 가능한 매개 변수를 사용하여 **Ansible** 플레이북을 설정하는 방법에 대해 알아봅니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.

참고

스토리지 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다.

- 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 추가하는 경우). 따라서 데이터 손실을 방지하기 위해 플레이북에서 특정 디스크 이름 사용을 권장하지 않습니다.

2. 선택 사항입니다. 플레이북 구문을 확인합니다.

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

관련 자료

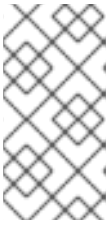
- **RAID**에 대한 자세한 내용은 **RAID** 관리를 참조하십시오.
- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.12. 스토리지 시스템 역할을 사용하여 RAID로 LVM 풀 구성

스토리지 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**에서 **RAID**를 사용하여 **LVM** 풀을 구성할 수 있습니다. 이 섹션에서는 **RAID**로 **LVM** 풀을 구성하는 데 사용 가능한 매개 변수를 사용하여 **Ansible** 플레이북을 설정하는 방법에 대해 알아봅니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.



참고

스토리지 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다**.**

- 플레이북을 실행할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다**.**

- 스토리지 시스템 역할을 사용하여 **RAID**로 **LVM** 풀을 구성할 시스템을 자세히 설명하는 인벤토리 파일이 있습니다**.**

절차

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

**RAID**를 사용하여 **LVM** 풀을 생성하려면 **raid_level** 매개 변수를 사용하여 **RAID** 유형을 지정해야 합니다**.**

2. 선택 사항입니다**.** 플레이북 구문을 확인합니다**.**

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

3. 인벤토리 파일에서 플레이북을 실행합니다.

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

관련 자료

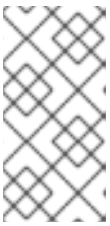
- **RAID**에 대한 자세한 내용은 **RAID** 관리를 참조하십시오.
- 스토리지 시스템 역할에 사용된 매개변수에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.storage/README.md** 파일을 참조하십시오.

## 13.13. 스토리지 역할을 사용하여 LUKS 암호화된 볼륨 생성

**Storage** 역할을 사용하여 **Ansible** 플레이북을 실행하여 **LUKS**로 암호화된 볼륨을 생성하고 구성할 수 있습니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.



참고

볼륨을 생성하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다.

- **Ansible** 컨트롤러에 **rhel-system-roles** 패키지가 설치되어 있어야 합니다.
- 스토리지 시스템 역할을 사용하여 **LUKS** 암호화된 볼륨을 배포하려는 시스템을 자세히 설명하는 인벤토리 파일이 있습니다.

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
```

관련 자료

- LUKS를 사용하여 블록 장치 암호화
- **/usr/share/ansible/roles/rhel-system-roles.storage/README.md file**

## 13.14. 관련 자료

- 자세한 내용은 **rhel-system-roles** 패키지를 설치하고 다음 디렉터리를 참조하십시오.
  - **/usr/share/doc/rhel-system-roles-*VERSION*/storage/**
  - **/usr/share/ansible/roles/rhel-system-roles.storage/**

# 14장. RHEL 시스템 역할을 사용하여 시간 동기화 구성

시간 동기화 **RHEL** 시스템 역할을 사용하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**의 여러 대상 머신에서 시간 동기화를 관리할 수 있습니다.

## 14.1. 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.

## 14.2. 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 않았더라도 손실됩니다. **timesync_ntp_provider** 변수가 정의되지 않은 경우 보존된 유일한 설정은 공급자 선택입니다.

사전 요구 사항

- 플레이북을 실행할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.

참고

**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**.**

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
```

## 14.3. 시간 동기화 시스템 역할 변수

다음 변수를 시간 동기화 역할에 전달할 수 있습니다**.**

- **timesync_ntp_servers:**

| 역할 변수 설정 | Description |
| --- | --- |
| 호스트 이름: host.example.com | 서버의 호스트 이름 또는 주소 |
| minpoll: *번호* | 최소 폴링 간격. 기본값: 6 |
| maxpoll: *number* | 최대 폴링 간격. 기본값: 10 |
| iburst: yes | 빠른 초기 동기화를 활성화하는 플래그. 기본값: no |
| 풀: yes | 호스트 이름의 확인된 각 주소가 별도의 NTP 서버임을 나타내는 플래그입니다. 기본값: no |

관련 자료

- 시간 동기화 역할 변수에 대한 자세한 참조는 **rhel-system-roles** 패키지를 설치하고 **/usr/share/doc/rhel-system-roles-VERSION/timesync** 디렉터리에 **README.md** 파일을 참조하십시오**.**

# 15장. RHEL 시스템 역할을 사용하여 성능 모니터링

시스템 관리자는 **Ansible Automation Platform** 제어 노드와 함께 **Metrics RHEL** 시스템 역할을 사용하여 시스템의 성능을 모니터링할 수 있습니다.

## 15.1. 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 **15.1.** 지표 시스템 역할 변수

| 역할 변수 | Description | 사용 예 |
|---|---|---|
| metrics_monitored_hosts | 대상 호스트에서 분석할 원격 호스트 목록입니다. 이러한 호스트에는 대상 호스트에 기록된 메트릭이 있으므로 각 호스트의 **/var/log** 아래에 디스크 공간이 충분히 있는지 확인합니다. | **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 false로 설정합니다. | **metrics_graph_service: false** |
| metrics_query_service | **redis** 를 통해 기록된 **pcp** 지표를 쿼리하기 위해 시계열 쿼리 서비스로 호스트를 설정할 수 있는 부울 플래그입니다. 기본적으로 false로 설정합니다. | **metrics_query_service: false** |
| metrics_provider | 지표를 제공하는 데 사용할 지표 수집기를 지정합니다. 현재는 **pcp** 가 지원되는 유일한 지표 프로바이더입니다. | **metrics_provider: "pcp"** |

참고

**metrics_connections** 에 사용되는 매개 변수 및 지표 시스템 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.metrics/README.md** 파일을 참조하십시오**.**

## **15.2.** 시각화로 로컬 시스템을 모니터링하기 위해 메트릭 시스템 역할을 사용

이 절차에서는 **grafana** 를 통해 데이터 시각화를 동시에 프로비저닝하는 동시에 **Metrics RHEL** 시스템 역할을 사용하여 로컬 시스템을 모니터링하는 방법을 설명합니다**.**

사전 요구 사항

- 모니터링할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다**.**
- 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 내용으로 **Ansible** 플레이북을 생성합니다**.**

```
---
- hosts: localhost
  vars:
    metrics_graph_service: yes
  roles:
    - rhel-system-roles.metrics
```

3. **Ansible** 플레이북을 실행합니다**.**

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

참고

**metrics_graph_service** 부울이 **value="yes"**로 설정되었으므로 **grafana** 는 데이터 소스로 추가된 **pcp** 를 사용하여 자동으로 설치 및 프로비저닝됩니다.

4. 시스템에서 수집 중인 지표의 시각화를 보려면 **Grafana** 웹 **UI** 액세스에 설명된 대로 **grafana** 웹 인터페이스에 액세스합니다.

## 15.3. 지표 시스템 역할을 사용하여 자신을 모니터링하기 위해 개별 시스템 세트를 설정

이 절차에서는 메트릭을 사용하여 모니터할 시스템 제품군을 설정하는 데 **Metrics** 시스템 역할을 사용하는 방법을 설명합니다.

사전 요구 사항

- 플레이북을 실행하는 데 사용할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
- 플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1. 플레이북을 통해 모니터링할 시스템의 이름 또는 **IP**를 괄호로 묶은 식별 그룹 이름으로 **/etc/ansible/hosts Ansible** 인벤토리 파일에 추가합니다.

```
[remotes]
webserver.example.com
database.example.com
```

2. 다음 내용으로 **Ansible** 플레이북을 생성합니다.

```
---
- hosts: remotes
  vars:
    metrics_retention_days: 0
  roles:
    - rhel-system-roles.metrics
```

3. **Ansible** 플레이북을 실행합니다.

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

**15.4.** 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

- 플레이북을 실행하는 데 사용할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
- 플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1. 다음 내용으로 **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** 플레이북을 실행합니다.

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

참고

**metrics_graph_service** 및 **metrics_query_service** 부울은 **value="yes"**로 설정되어 있으므로 **grafana** 는 **pcp** 데이터 기록이 **redis** 에 인덱싱된 데이터 소스로 추가된 **pcp** 쿼리 언어를 자동으로 설치 및 프로비저닝하므로 **pcp** 쿼리 언어를 복잡한 데이터 쿼리에 사용할 수 있습니다.

3. 시스템에서 중앙에서 수집 중인 지표의 그래픽 표시를 보고 데이터를 쿼리하려면 **Grafana** 웹 **UI** 액세스에 설명된 대로 **grafana** 웹 인터페이스에 액세스합니다.

## 15.5. METRICS 시스템 역할을 사용하여 시스템을 모니터링하는 동안 인증 설정

**PCP**는 **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

- 플레이북을 실행하는 데 사용할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
- 플레이북을 실행하는 데 사용할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.

절차

1. 인증을 설정하려는 **Ansible** 플레이북에 다음 변수를 포함합니다.

```
---
 vars:
   metrics_username: your_username
   metrics_password: your_password
```

2. **Ansible** 플레이북을 실행합니다.

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

검증 단계

- **the sasl** 구성을 확인합니다.

```
# pminfo -f -h "pcp://127.0.0.1?username=your_username" disk.dev.read
Password:
disk.dev.read
inst [0 or "sda"] value 19540
```

### 15.6. 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

- 모니터링할 시스템에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
- 모니터링할 시스템에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- **Microsoft SQL Server for Red Hat Enterprise Linux**를 설치하고 **SQL** 서버에 '신뢰할 수 있는' 연결을 구축했습니다.
- **Red Hat Enterprise Linux**용 **SQL Server**용 **Microsoft ODBC** 드라이버를 설치했습니다.

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_sql: yes
```

3. **Ansible** 플레이북을 실행합니다.

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

검증 단계

- **pcp** 명령을 사용하여**mssql(SQL** 서버 **PMDA** 에이전트**)**이 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
```

관련 자료

- **Microsoft SQL Server**용 **Performance Co-Pilot** 사용에 대한 자세한 내용은 이 **Red Hat Developers** 블로그 게시물을 참조하십시오.

# 16장. RHEL 시스템 역할을 기록하는 터미널 세션을 사용하여 세션 레코딩 시스템 구성

터미널 세션이 **RHEL** 시스템 역할을 기록하면 **Red Hat Ansible Automation Platform**을 사용하여 **RHEL**에서 터미널 세션 레코딩 시스템을 구성할 수 있습니다.

## 16.1. 터미널 세션 기록 시스템 역할

**RHEL** 시스템 역할을 기록하는 터미널 세션을 사용하여 **RHEL**에서 터미널 세션 레코딩을 위해 **RHEL** 시스템을 구성할 수 있습니다. **tlog** 패키지 및 관련 웹 콘솔 세션 플레이어를 통해 사용자 터미널 세션을 기록하고 재생할 수 있습니다.

**SSSD** 서비스를 통해 사용자 또는 사용자 그룹별로 기록이 수행되도록 구성할 수 있습니다. 모든 터미널 입력 및 출력은 캡처되어 시스템 저널의 텍스트 기반 형식으로 저장됩니다.

관련 자료

- **RHEL**의 세션 기록에 대한 자세한 내용은 기록 세션을참조하십시오.

## 16.2. 시스템 역할을 기록하는 터미널 세션의 구성 요소 및 매개변수

세션 기록 솔루션은 다음 구성 요소로 구성됩니다.

- **tlog** 유틸리티
- **SSSD**(시스템 보안 서비스 데몬)
- 선택 사항: 웹 콘솔 인터페이스

**RHEL** 시스템 역할을 기록하는 터미널 세션에 사용되는 매개변수는 다음과 같습니다.

| 역할 변수 | Description |
|---|---|
| tlog_use_sssd (기본값: yes) | 기록된 사용자 또는 그룹을 관리하는 기본 방법인 SSSD를 사용하여 세션 기록 구성 |

| 역할 변수 | Description |
| --- | --- |
| tlog_scope_sssd (default: none) | SSSD 기록 범위 설정 - all / some / none |
| tlog_users_sssd (default: []) | 기록할 사용자 목록 |
| tlog_groups_sssd (default: []) | 기록할 그룹 목록 |

- **tlog** 에 사용되는 매개 변수 및 시스템 역할을 기록하는 터미널 세션 역할에 대한 자세한 내용은 **/usr/share/ansible/roles/rhel-system-roles.tlog/README.md** 파일을 참조하십시오**.**

## 16.3. RHEL 시스템 역할을 기록하는 터미널 세션 배포

다음 단계에 따라 데이터를 **systemd** 저널에 기록하도록 **Ansible** 플레이북을 준비하고 적용하여 **RHEL** 시스템을 구성합니다**.**

사전 요구 사항

- 제어 노드에서 시스템 역할을 기록하는 터미널 세션의 대상 시스템으로 액세스할 **SSH** 키를 설정했습니다**.**
- **Ansible Engine**이 다른 시스템을 구성하는 시스템인 하나의 제어 노드가 있습니다**.**
- 플레이북을 실행할 제어 노드에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다**.**
- 플레이북을 실행할 제어 노드에 **rhel-system-roles** 패키지가 설치되어 있습니다**.**
- 터미널 세션 레코딩 시스템 역할을 구성할 시스템이 하나 이상 있습니다**. tlog** 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다**.**

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

다음과 같습니다.

- **tlog_scope_sssd:**
  - 일부에서는 일부 또는 없음이 아닌 특정 사용자와 그룹만 기록합니다.
- **tlog_users_sssd:**
  - **recordeduser** 는 에서 세션을 기록할 사용자를 지정합니다. 이 명령은 사용자를 추가하지 않습니다. 사용자를 직접 설정해야 합니다.

2. 원하는 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 레코딩 역할을 설치합니다. 또한 사용자가 정의한 사용자와 그룹이 사용할 수 있는 **SSSD** 구성 드롭 파일을 생성합니다. **SSSD**는 이러한 사용자와 그룹을 구문 분석하고 읽어 **tlog** 세션을 쉘 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

**SSSD** 구성 드롭 파일이 시스템에 생성되었는지 확인하려면 다음 단계를 수행합니다.

1. **SSSD** 구성 드롭 파일이 생성된 폴더로 이동합니다.

```
# cd /etc/sssd/conf.d
```

2. 파일 내용을 확인합니다.

```
# cat /etc/sssd/conf.d/sssd-session-recording.conf
```

파일에 플레이북에서 설정한 매개 변수가 포함되어 있음을 확인할 수 있습니다.

## 16.4. 그룹 또는 사용자 목록을 제외하기 위해 RHEL 시스템 역할을 기록하는 터미널 세션 배포

**RHEL**에서 시스템 역할을 기록하는 터미널 세션 레코딩 구성 옵션을 사용할 수 있습니다. **exclude_users** 및 **exclude_groups.** 다음 단계에 따라 사용자 또는 그룹이 **systemd** 저널에 기록되고 기록되지 않도록 **RHEL** 시스템을 구성하도록 **Ansible** 플레이북을 준비하고 적용합니다.

사전 요구 사항

- 제어 노드에서 시스템 역할을 기록하는 터미널 세션을 구성하려는 대상 시스템으로 **SSH** 키를 설정했습니다.
- **Red Hat Ansible Engine**이 다른 시스템을 구성하는 하나의 제어 노드가 있습니다.
- 플레이북을 실행할 제어 노드에 **Red Hat Ansible Engine**이 설치되어 있어야 합니다.
- 제어 노드에 **rhel-system-roles** 패키지가 설치되어 있습니다.
- 터미널 세션 레코딩 시스템 역할을 구성하려는 시스템이 하나 이상 있습니다.

  **tlog** 솔루션을 배포하려는 시스템에 **Red Hat Ansible Automation Platform**을 설치할 필요가 없습니다.

절차

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

```
---
- name: Deploy session recording excluding users and groups
  hosts: all
  vars:
    tlog_scope_sssd: all
    tlog_exclude_users_sssd:
      - jeff
      - james
    tlog_exclude_groups_sssd:
      - admins

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

다음과 같습니다.

- **tlog_scope_sssd:**
  - **all :** 모든 사용자와 그룹을 기록하도록 지정합니다.
- **tlog_exclude_users_sssd:**
  - 사용자 이름**:** 세션 녹화에서 제외하려는 사용자의 사용자 이름을 지정합니다.
- **tlog_exclude_groups_sssd:**
  - **admins** 는 세션 녹화에서 제외하려는 그룹을 지정합니다.

2. 선택적으로 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 사용자가 지정한 시스템에 **tlog** 패키지를 설치합니다. 또한 사용자가 제외한 것을 제외하고 사용자 및 그룹에서 사용할 수 있는 **/etc/sssd/conf.d/ssd-session-recording.conf SSSD** 구성 드롭 파일을 생성합니다. **SSSD**는 이러한 사용자와 그룹을 구문 분석하고 읽고 쉘 사용자로 **tlog** 세션을 겹칩니다. 또한 **cockpit** 패키지가 시스템에 설치된 경우 플레이북은 웹 콘솔 인터페이스에서 녹화를 보고 플레이할 수 있는 **Cockpit** 모듈인 **cockpit-session-recording** 패키지도 설치합니다.

참고

**exclude_users** 목록에 나열된 사용자 또는 **exclude_ groups** 목록에 있는 그룹의 멤버인 사용자에 대한 세션을 기록할 수 없습니다.

검증 단계

**SSSD** 구성 드롭 파일이 시스템에 생성되었는지 확인하려면 다음 단계를 수행합니다.

1. **SSSD** 구성 드롭 파일이 생성된 폴더로 이동합니다.

```
# cd /etc/sssd/conf.d
```

2. 파일 내용을 확인합니다.

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

파일에 플레이북에서 설정한 매개 변수가 포함되어 있음을 확인할 수 있습니다.

관련 자료

- **/usr/share/doc/rhel-system-roles-VERSION/tlog/** 및 **/usr/share/ansible/roles/rhel-system-roles.tlog/** 디렉토리를 참조하십시오.
- **16.5**절. **“CLI**에서 배포된 터미널 세션 기록 시스템 역할을 사용하여 세션 기록” 참조하십시오.

## 16.5. CLI에서 배포된 터미널 세션 기록 시스템 역할을 사용하여 세션 기록

지정한 시스템에 터미널 세션 레코딩 시스템 역할을 배포하면 **CLI(**명령줄 인터페이스**)**를 사용하여 사용자 터미널 세션을 기록할 수 있습니다**.**

사전 요구 사항

- 대상 시스템에 터미널 세션 레코딩 시스템 역할을 배포했습니다**.**
- **SSSD** 구성 드롭 파일은 **/etc/sssd/conf.d** 파일에 생성되었습니다**.**

절차

1. 사용자를 생성하고 이 사용자에 대한 암호를 할당합니다**.**

```
# useradd recordeduser
# passwd recordeduser
```

2. 방금 만든 사용자로 시스템에 다시 로그인합니다**.**

```
# ssh recordeduser@localhost
```

3. 시스템이 **yes** 또는 **no**를 입력하여 인증하라는 메시지가 표시되면 **"yes"**를 입력합니다**.**
4. ***recordeduser****의* 암호를 삽입합니다**.**

   세션이 기록되고 있음을 알리는 메시지 메시지가 표시됩니다**.**

```
ATTENTION! Your session is being recorded!
```

5. 세션 기록이 완료되면 다음을 입력합니다**.**

```
# exit
```

   시스템은 사용자로부터 로그아웃하고 **localhost**와의 연결을 종료합니다**.**

결과적으로 사용자 세션이 기록되고 저장되며 저널을 사용하여 수행할 수 있습니다.

검증 단계

저널에서 기록된 세션을 보려면 다음 단계를 수행하십시오.

1. 다음 명령을 실행합니다.

```
# journalctl -o verbose -r
```

2. **tlog-rec** 기록된 저널 항목의 **MESSAGE** 필드를 검색합니다.

```
# journalctl -xel _EXE=/usr/bin/tlog-rec-session
```

## 16.6. CLI를 사용하여 기록된 세션 조회

**CLI(**명령줄 인터페이스**)**를 사용하여 저널에서 사용자 세션 기록을 수행할 수 있습니다.

사전 요구 사항

- 사용자 세션을 기록했습니다. **CLI**에서 배포된 터미널 세션 기록 시스템 역할을 사용하여세션 기록 참조

절차

1. **CLI** 터미널에서 사용자 세션 기록을 수행합니다.

```
# journalctl -o verbose -r
```

2. **tlog** 기록을 검색합니다.

```
$ /tlog-rec
```

다음과 같은 세부 정보를 볼 수 있습니다.

- 사용자 세션 기록의 사용자 이름
- **out_txt** 필드, 기록된 세션의 원시 출력 인코딩
- 식별자 번호 **TLOG_REC=*ID_number***

3. 식별자 번호 **TLOG_REC=*ID_number*** 를 복사합니다.

4. 식별자 번호 **TLOG_REC=*ID_number*** *를 사용하여 기록을 재생합니다*.

```
# tlog-play -r journal -M TLOG_REC=ID_number
```

결과적으로 사용자 세션 기록 터미널 출력이 다시 재생되는 것을 확인할 수 있습니다.