Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform 업그레이드 및 마이그레이션 가이드

Ansible Automation Platform의 레거시 배포 업그레이드 및 마이그레이션

Last Updated: 2024-06-13

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform 업그레이드 및 마이그레이션 가이드

Ansible Automation Platform의 레거시 배포 업그레이드 및 마이그레이션

법적 공지

초록

이 가이드에서는 최신 버전의 Ansible Automation Platform으로 업그레이드하고 기존 가상 환경을 자동화 실행 환경으로 마이그레이션하는 방법을 보여줍니다.

차례

RED HAT 문서에 관한 피드백 제공

이 문서를 개선하기 위한 제안이 있거나 오류를 찾을 수 있는 경우 https://access.redhat.com 에서 기술 지원에 문의하여 **docs-product** 구성 요소를 사용하여 Ansible Automation Platform Jira 프로젝트에 문제를 생성하십시오.

1장. RED HAT ANSIBLE AUTOMATION PLATFORM 2.4 업그레이드

인벤토리를 설정하고 설치 스크립트를 실행하여 Red Hat Ansible Automation Platform 2.4로 업그레이드합니다. 그런 다음 Ansible이 배포를 2.4로 업그레이드합니다. Ansible Automation Platform 2.0 이하에서 업그레이드하려는 경우 2.4와의 호환성을 위해 Ansible 콘텐츠를 마이그레이션해야 합니다.

1.1. ANSIBLE AUTOMATION PLATFORM 업그레이드

Ansible Automation Platform 2.1 이상에서 버전 2.4로 업그레이드하려면 설치 패키지를 다운로드한 다음 다음 단계를 수행해야 합니다.

- 설치 환경과 일치하도록 인벤토리를 설정합니다.
- 현재 Ansible Automation Platform 설치를 통해 2.4 설치 프로그램을 실행합니다.

추가 리소스

- Red Hat Ansible Automation Platform 2.4로 업그레이드

1.2. ANSIBLE AUTOMATION PLATFORM 레거시 업그레이드

Ansible Automation Platform 2.0 이하에서 버전 2.4로 업그레이드하려면 호환성을 위해 Ansible 콘텐츠를 마이그레이션해야 합니다.

다음 단계에서는 레거시 업그레이드 프로세스에 대한 개요를 제공합니다.

- **awx-manage** 명령을 사용하여 사용자 지정 가상 환경을 자동화 실행 환경으로 복제합니다.
- 노드가 최신 자동화 메시 기능과 호환되도록 병렬 업그레이드를 수행하여 격리된 레거시 노드에서 실행 노드로 데이터를 마이그레이션합니다.
- 새 자동화 허브 API 토큰을 가져오거나 생성합니다.
- **ansible-core** 2.15와의 호환성을 위해 정규화된 컬렉션 이름(FQCN)을 포함하도록 Ansible 콘텐츠를 재구성하십시오.

추가 리소스

- 가상 환경을 자동화 실행 환경으로 마이그레이션
- 격리된 노드를 실행 노드로 마이그레이션
- 마이그레이션 Ansible 콘텐츠

2장. RED HAT ANSIBLE AUTOMATION PLATFORM 2.4로 업그레이드

Red Hat Ansible Automation Platform을 업그레이드하려면 계획 정보를 검토하여 업그레이드를 시작하십시오. 그런 다음 원하는 버전의 Ansible Automation Platform 설치 프로그램을 다운로드하고, 설치 번들에 인벤토리 파일을 구성하여 환경을 반영한 다음 설치 프로그램을 실행할 수 있습니다.

2.1. ANSIBLE AUTOMATION PLATFORM 업그레이드 계획

업그레이드 프로세스를 시작하기 전에 다음 고려 사항을 검토하여 Ansible Automation Platform 배포를 계획하고 준비합니다.

자동화 컨트롤러

- 이전 버전의 유효한 라이센스가 있더라도 최신 버전의 자동화 컨트롤러로 업그레이드할 때 인증 정보 또는 서브스크립션 매니페스트를 제공해야 합니다.
- Red Hat Enterprise Linux 및 자동화 컨트롤러를 업그레이드해야 하는 경우 먼저 자동화 컨트롤러 데이터를 백업하고 복원해야 합니다.
- 클러스터형 업그레이드는 업그레이드하기 전에 인스턴스 및 인스턴스 그룹에 특별히 주의해야 합니다.

추가 리소스

- 서브스크립션 가져오기
- 백업 및 복원
- 클러스터링

자동화 허브

- Ansible Automation Platform 2.4로 업그레이드할 때 기존 자동화 허브 API 토큰을 추가하거나 새 토큰을 생성하고 기존 토큰을 무효화할 수 있습니다.

추가 리소스

- 인벤토리 파일 설정

이벤트 기반 **Ansible** 컨트롤러

- 현재 Event-Driven Ansible 컨트롤러를 실행하고 Ansible Automation Platform 2.4로 업그레이드할 때 배포할 예정인 경우 업그레이드 프로세스가 완료된 후 새 활성화만 실행하도록 업그레이드하기 전에 모든 Event-Driven Ansible 활성화를 비활성화하는 것이 좋습니다. 이렇게 하면 이전 버전의 활성화를 실행하는 분리된 컨테이너의 가능성을 방지할 수 있습니다.

2.2. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 선택 및 가져오기

Red Hat Enterprise Linux 환경 인터넷 연결을 기반으로 필요한 Red Hat Ansible Automation Platform 설치 프로그램을 선택합니다. 다음 시나리오를 검토하고 요구 사항을 충족하는 Red Hat Ansible Automation Platform 설치 프로그램을 결정합니다.

참고

Red Hat 고객 포털에서 Red Hat Ansible Automation Platform 설치 프로그램에 액세스하려면 유효한 Red Hat 고객 계정이 필요합니다.

인터넷으로 설치

Red Hat Enterprise Linux 환경이 인터넷에 연결된 경우 Red Hat Ansible Automation Platform 설치 프로그램을 선택합니다. 인터넷 액세스가 포함된 설치는 최신 필수 리포지토리, 패키지 및 종속성을 검색합니다. 다음 방법 중 하나를 선택하여 Ansible Automation Platform 설치 프로그램을 설정합니다.

tarball 설치

1. Red Hat Ansible Automation Platform 다운로드 페이지로 이동합니다.
2. **Ansible Automation Platform <latest-version> 설정에 대해지금 다운로드를 클릭합니다**.
3. 파일을 추출합니다.

```
$ tar xvzf ansible-automation-platform-setup-<latest-version>.tar.gz
```

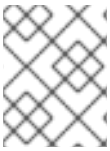
RPM 설치

1. Ansible Automation Platform 설치 프로그램 패키지 설치
   V.2.4 for RHEL 8 for x86_64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   V.2.4 for RHEL 9 for x86-64

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms
ansible-automation-platform-installer
```



참고

리포지토리가 기본적으로 비활성화되어 있으므로 **dnf install** 을 사용하면 리포지터리를 사용할 수 있습니다.

RPM 설치 프로그램을 사용하면 파일이 **/opt/ansible-automation-platform/installer** 디렉터리에 배치됩니다.

인터넷 액세스없이 설치

인터넷에 액세스할 수 없거나 온라인 리포지토리와 별도의 구성 요소 및 종속 항목을 설치하지 않으려는 경우 Red Hat Ansible Automation Platform **Bundle** 설치 관리자를 사용합니다. Red Hat Enterprise Linux 리포지토리에 대한 액세스는 여전히 필요합니다. 기타 모든 종속 항목은 tar 아카이브에 포함되어 있습니다.

1. Red Hat Ansible Automation Platform 다운로드 페이지로 이동합니다.
2. **Ansible Automation Platform <latest-version> 설치 번들로지금 다운로드를** 클릭합니다.
3. 파일을 추출합니다.

```
$ tar xvzf ansible-automation-platform-setup-bundle-<latest-version>.tar.gz
```

2.3. 인벤토리 파일 설정

Red Hat Ansible Automation Platform 설치를 업그레이드하기 전에 원하는 구성과 일치하도록 **인벤토리** 파일을 편집합니다. 기존 Ansible Automation Platform 배포에서 동일한 매개변수를 유지하거나 환경에 대한 변경 사항과 일치하도록 매개변수를 수정할 수 있습니다.

절차

1. 설치 프로그램 디렉터리로 이동합니다.

   번들 설치

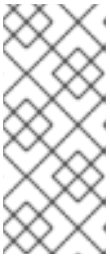
   ```
   $ cd ansible-automation-platform-setup-bundle-2.4-1-x86_64
   ```

   온라인 설치 프로그램

   ```
   $ cd ansible-automation-platform-setup-2.4-1
   ```

2. 편집할 **인벤토리** 파일을 엽니다.

3. **인벤토리** 파일을 수정하여 새 노드를 프로비저닝하고, 노드 또는 그룹을 프로비저닝 해제하고, 자동화 허브 API 토큰을 가져오거나 생성합니다.
   환경을 변경하지 않는 경우 기존 Ansible Automation Platform 2.1 설치의 동일한 **인벤토리** 파일을 사용할 수 있습니다.



참고

[automationhub] 및 **[automationcontroller]** 호스트에 연결할 수 있는 IP 주소 또는 FQDN(정규화된 도메인 이름)을 제공하여 사용자가 다른 노드에서 Ansible 자동화 허브에서 콘텐츠를 동기화하고 설치할 수 있도록 합니다. **localhost** 를 사용하지 마십시오. **localhost** 를 사용하는 경우 업그레이드는 preflight 검사의 일부로 중지됩니다.

클러스터에서 새 노드 프로비저닝

- 다음과 같이 **인벤토리** 파일의 기존 노드와 함께 새 노드를 추가합니다.

```
[controller]
clusternode1.example.com
clusternode2.example.com
clusternode3.example.com

[all:vars]
admin_password='password'

pg_host=''
pg_port=''
```

```
pg_database='<database_name>'
pg_username='<your_username>'
pg_password='<your_password>'
```

클러스터에서 노드 또는 그룹 프로비저닝 해제

- **inventory** 파일 내의 노드 또는 그룹에 **node_state-deprovision** 를 추가합니다.

API 토큰 가져오기 및 생성

Red Hat Ansible Automation Platform 2.0 이상에서 Red Hat Ansible Automation Platform 2.1 이상으로 업그레이드하는 경우 기존 자동화 허브 API 토큰을 사용하거나 새 토큰을 생성할 수 있습니다. 인벤토리 파일에서 Red Hat Ansible Automation Platform 설치 스크립트 **setup.sh** 를 실행하기 전에 다음 필드 중 하나를 편집합니다.

- 다음과 같이 **automationhub_api_token** 플래그를 사용하여 기존 API 토큰을 가져옵니다.

```
automationhub_api_token=<api_token>
```

- 다음과 같이 새 API 토큰을 생성하고 **generate_automationhub_token** 플래그를 사용하여 기존 토큰을 무효화합니다.

```
generate_automationhub_token=True
```

추가 리소스

- Red Hat Ansible Automation Platform 설치 가이드
- 개별 노드 또는 인스턴스 그룹 프로비저닝 해제

2.4. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 설정 스크립트 실행

인벤토리 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

```
$ ./setup.sh
```

설치가 시작됩니다.

3장. 자동화 실행 환경으로 마이그레이션

3.1. 자동화 실행 환경으로 업그레이드하는 이유는 무엇입니까?

Red Hat Ansible Automation Platform 2.4에는 자동화 실행 환경이 도입되었습니다. 자동화 실행 환경은 단일 컨테이너 내에서 Ansible 자동화를 실행하는 데 필요한 모든 것을 포함하여 Ansible을 더 쉽게 관리할 수 있는 컨테이너 이미지입니다. 자동화 실행 환경은 다음과 같습니다.

- RHEL UBI 8
- ansible-core 2.14 이상
- Python 3.9 이상
- 모든 Ansible 콘텐츠 컬렉션
- 컬렉션 python 또는 바이너리 종속 항목

Ansible은 이러한 요소를 포함하여 플랫폼 관리자에게 자동화를 실행하는 환경을 정의, 빌드 및 배포하는 표준화된 방법을 제공합니다.

새로운 자동화 실행 환경으로 인해 더 이상 관리자가 사용자 지정 플러그인 및 자동화 콘텐츠를 생성할 필요가 없습니다. 이제 관리자는 콘텐츠를 생성하기 위해 더 적은 시간에 더 작은 자동화 실행 환경을 가동할 수 있습니다.

이제 모든 사용자 지정 종속성은 관리 및 배포 단계가 아닌 개발 단계에서 정의됩니다. 컨트롤 플레인과 분리하면 환경 전반에서 개발 사이클, 확장성, 신뢰성 및 이식성이 빨라집니다. 자동화 실행 환경을 통해 Ansible Automation Platform은 분산 아키텍처로 이동할 수 있으므로 관리자가 조직 전체에서 자동화를 확장할 수 있습니다.

3.2. 레거시 VENVS를 자동화 실행 환경으로 마이그레이션하는 정보

이전 버전의 자동화 컨트롤러에서 버전 4.0 이상으로 업그레이드할 때 컨트롤러는 조직, 인벤토리, 작업 템플릿과 관련된 이전 버전의 가상 환경을 감지하고 새 자동화 실행 환경 모델로 마이그레이션하도록 지시할 수 있습니다. 자동화 컨트롤러를 새로 설치하면 설치 중에 virtualenvs 두 개가 생성됩니다. 하나는 컨트롤러를 실행하고 다른 하나는 Ansible을 실행합니다. 기존 가상 환경과 마찬가지로 자동화 실행 환경을 사용하면 컨트롤러가 안정적인 환경에서 실행할 수 있으며, 플레이북을 실행하는 데 필요한 경우 자동화 실행 환경에 모듈을 추가하거나 업데이트할 수 있습니다.

자동화 실행 환경에서 새 자동화 실행 환경으로 마이그레이션하여 이전 사용자 지정 가상 환경에서 설치를 복제할 수 있습니다. 이 섹션에서 **awx-manage** 명령을 사용하여 다음을 수행합니다.

- 현재 사용자 지정 가상 환경 및 해당 경로 목록(**list_custom_venvs**)
- 특정 사용자 지정 가상 환경(**custom_venv_associations**)을 사용하는 리소스 보기
- 자동화 실행 환경으로 마이그레이션하는 데 사용할 수 있는 형식으로 특정 사용자 지정 가상 환경을 내보냅니다(**export_custom_venv**).

다음 워크플로는 **awx-manage** 명령을 사용하여 레거시 venvs에서 자동화 실행 환경으로 마이그레이션하는 방법을 설명합니다.

3.3. 가상 환경을 자동화 실행 환경으로 마이그레이션

Red Hat Ansible Automation Platform 2.0 및 자동화 컨트롤러 4.0으로 업그레이드되면 다음 섹션을 사용하여 마이그레이션 프로세스의 추가 단계를 지원합니다.

3.3.1. 사용자 정의 가상 환경 나열

awx-manage 명령을 사용하여 자동화 컨트롤러 인스턴스의 가상 환경을 나열할 수 있습니다.

절차

1. 자동화 컨트롤러 인스턴스에 SSH를 사용하여 다음을 실행합니다.

```
$ awx-manage list_custom_venvs
```

검색된 가상 환경 목록이 표시됩니다.

```
# Discovered virtual environments:
/var/lib/awx/venv/testing
/var/lib/venv/new_env

To export the contents of a virtual environment, re-run while supplying the path as an argument:
awx-manage export_custom_venv /path/to/venv
```

3.3.2. 사용자 정의 가상 환경과 연결된 오브젝트 보기

awx-manage 명령을 사용하여 사용자 지정 가상 환경과 관련된 조직, 작업 및 인벤토리 소스를 확인합니다.

절차

1. 자동화 컨트롤러 인스턴스에 SSH를 사용하여 다음을 실행합니다.

```
$ awx-manage custom_venv_associations /path/to/venv
```

연결된 오브젝트 목록이 나타납니다.

```
inventory_sources:
- id: 15
  name: celery
job_templates:
- id: 9
  name: Demo Job Template @ 2:40:47 PM
- id: 13
  name: elephant
organizations
- id: 3
  name: alternating_bongo_meow
- id: 1
  name: Default
projects: []
```

3.3.3. 내보낼 사용자 정의 가상 환경 선택

awx-manage export_custom_venv 명령을 사용하여 내보낼 사용자 지정 가상 환경을 선택합니다.

절차

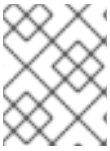
1. 자동화 컨트롤러 인스턴스에 SSH를 사용하여 다음을 실행합니다.

```
$ awx-manage export_custom_venv /path/to/venv
```

이 명령의 출력은 지정된 가상 환경에 있는 항목의 **pip 동결** 을 표시합니다. 이 정보는 Ansible Builder가 새 자동화 실행 환경 이미지를 생성하는 데 사용할 **requirements.txt** 파일에 복사할 수 있습니다.

```
numpy==1.20.2
pandas==1.2.4
python-dateutil==2.8.1
pytz==2021.1
six==1.16.0

To list all available custom virtual environments run:
awx-manage list_custom_venvs
```



참고

출력을 줄이기 위해 **awx-manage list_custom_venvs** 를 실행할 때 **-q** 플래그를 전달합니다.

4장. 격리된 노드를 실행 노드로 마이그레이션

버전 1.x에서 Red Hat Ansible Automation Platform의 최신 버전으로 업그레이드하려면 플랫폼 관리자가 격리된 기존 노드에서 실행 노드로 데이터를 마이그레이션해야 합니다. 이 마이그레이션은 자동화 메시를 배포하는 데 필요합니다.

이 가이드에서는 병렬 마이그레이션을 수행하는 방법을 설명합니다. 이렇게 하면 마이그레이션 프로세스 중에 원래 자동화 환경의 데이터가 그대로 유지됩니다.

마이그레이션 프로세스에는 다음 단계가 포함됩니다.

1. 업그레이드 구성을 확인합니다.
2. 원래 인스턴스를 백업합니다.
3. 병렬 업그레이드를 위해 새 인스턴스를 배포합니다.
4. ansible 컨트롤러를 사용하여 새 인스턴스에서 인스턴스 그룹을 재생성합니다.
5. 원래 백업을 새 인스턴스로 복원합니다.
6. 실행 노드를 설정하고 인스턴스를 Red Hat Ansible Automation Platform 2.4로 업그레이드합니다.
7. 업그레이드된 컨트롤러 인스턴스를 구성합니다.

4.1. ANSIBLE AUTOMATION PLATFORM 업그레이드를 위한 사전 요구 사항

Ansible Automation Platform 업그레이드를 시작하기 전에 환경이 다음 노드 및 구성 요구 사항을 충족하는지 확인하십시오.

4.1.1. 노드 요구 사항

Ansible Automation Platform 업그레이드 프로세스와 관련된 노드에 다음 사양이 필요합니다.

- 컨트롤러 노드, 데이터베이스 노드, 실행 노드 및 홉 노드용 16GB RAM.
- 컨트롤러 노드, 데이터베이스 노드, 실행 노드 및 홉 노드의 CPU 4개
- 데이터베이스 노드의 150GB 이상의 디스크 공간
- 데이터베이스 이외의 노드의 경우 40GB 이상의 디스크 공간
- DHCP 예약은 무한 리스를 사용하여 고정 IP 주소로 클러스터를 배포합니다.
- 모든 노드의 DNS 레코드입니다.
- 모든 노드에 대해 설치된 Red Hat Enterprise Linux 8 이상 64비트(x86)입니다.
- 모든 노드에 대해 구성된 chrony입니다.
- 모든 콘텐츠 종속 항목에 대한 Python 3.9 이상

4.1.2. 자동화 컨트롤러 구성 요구사항

Ansible Automation Platform 업그레이드 프로세스를 진행하기 전에 다음 자동화 컨트롤러 구성이 필요합니다.

Chrony를 사용하여 NTP 서버 구성

클러스터의 각 Ansible Automation Platform 노드는 NTP 서버에 액세스할 수 있어야 합니다. **chronyd** 를 사용하여 시스템 시계를 NTP 서버와 동기화합니다. 이렇게 하면 노드 간 날짜 및 시간이 동기화되지 않는 경우 검증이 필요한 SSL 인증서를 사용하는 클러스터 노드가 실패하지 않습니다.

이는 업그레이드된 Ansible Automation Platform 클러스터에 사용된 모든 노드에 필요합니다.

1. **chrony** 를 설치합니다.

```
# dnf install chrony --assumeyes
```

2. 텍스트 편집기를 사용하여 **/etc/chrony.conf** 를 엽니다.

3. 공용 서버 풀 섹션을 찾아 적절한 NTP 서버 주소를 포함하도록 수정합니다. 하나의 서버만 필요하지만 3개의 서버를 사용하는 것이 좋습니다. 'iburst' 옵션을 추가하여 서버와 올바르게 동기화하는 데 걸리는 시간을 단축합니다.

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server <ntp-server-address> iburst
```

4. **/etc/chrony.conf** 파일에 변경 사항을 저장합니다.

5. 호스트를 시작하고 **chronyd** 데몬을 활성화합니다.

```
# systemctl --now enable chronyd.service
```

6. **chronyd** 데몬 상태를 확인합니다.

```
# systemctl status chronyd.service
```

모든 노드에 Red Hat 서브스크립션 연결

Red Hat Ansible Automation Platform을 사용하려면 모든 노드에 유효한 서브스크립션이 연결되어 있어야 합니다. 다음 명령을 실행하여 현재 노드에 Red Hat 서브스크립션이 있는지 확인할 수 있습니다.

```
# subscription-manager list --consumed
```

노드에 연결된 Red Hat 서브스크립션이 없는 경우 자세한 내용은 Ansible Automation Platform 서브스크립션 연결을 참조하십시오.

sudo 권한을 사용하여 root가 아닌 사용자 생성

Ansible Automation Platform을 업그레이드하기 전에 배포 프로세스에 sudo 권한이 있는 root 이외의 사용자를 생성하는 것이 좋습니다. 이 사용자는 다음을 위해 사용됩니다.

- SSH 연결.
- 설치 중 암호 없는 인증입니다.

- 권한 상승(sudo) 권한.

다음 예제에서는 **ansible** 을 사용하여 이 사용자의 이름을 지정합니다. 업그레이드된 Ansible Automation Platform 클러스터에 사용된 모든 노드에서 **ansible** 이라는 루트가 아닌 사용자를 생성하고 SSH 키를 생성합니다.

1. root가 아닌 사용자를 생성합니다.

```
# useradd ansible
```

2. 사용자 암호를 설정합니다.

```
# passwd ansible 1
Changing password for ansible.
Old Password:
New Password:
Retype New Password:
```

1 다른 이름을 사용하는 경우 1단계의 root가 아닌 사용자로 **ansible** 을 교체합니다.

3. 사용자로 **ssh** 키를 생성합니다.

```
$ ssh-keygen -t rsa
```

4. **sudo** 사용 시 암호를 요구하지 않도록 비활성화합니다.

```
# echo "ansible ALL=(ALL) NOPASSWD:ALL" | sudo tee -a /etc/sudoers.d/ansible
```

모든 노드에 SSH 키 복사

ansible 사용자가 생성된 상태에서 업그레이드된 Ansible Automation Platform 클러스터에 사용된 모든 노드에 **ssh** 키를 복사합니다. 이렇게 하면 Ansible Automation Platform 설치가 실행될 때 암호 없이 모든 노드에 **ssh** 를 실행할 수 있습니다.

```
$ ssh-copy-id ansible@node-1.example.com
```

참고

클라우드 공급자 내에서 실행하는 경우 모든 노드에서 **ansible** 사용자의 공개 키가 포함된 **~/.ssh/authorized_keys** 파일을 생성하고, 읽기 및 쓰기 액세스 권한(권한 600)만 소유자(Ansible)로만 권한을 **authorized_keys** 파일로 설정해야 할 수 있습니다.

방화벽 설정 구성

업그레이드된 Ansible Automation Platform 클러스터에 사용된 모든 노드에서 방화벽 설정을 구성하여 Ansible Automation Platform 업그레이드를 위해 적절한 서비스 및 포트에 액세스할 수 있습니다. Red Hat Enterprise Linux 8 이상의 경우 **firewalld** 데몬을 활성화하여 모든 노드에 필요한 액세스 권한을 활성화합니다.

1. **firewalld** 패키지를 설치합니다.

```
# dnf install firewalld --assumeyes
```

2. **firewalld** 서비스를 시작합니다.

```
# systemctl start firewalld
```

3. **firewalld** 서비스를 활성화합니다.

```
# systemctl enable --now firewalld
```

4.1.3. Ansible Automation Platform 구성 요구사항

Ansible Automation Platform 업그레이드 프로세스를 진행하기 전에 다음 Ansible Automation Platform 구성이 필요합니다.

실행 및 홉 노드에 대한 방화벽 설정 구성

Red Hat Ansible Automation Platform 인스턴스를 업그레이드한 후 메시 노드(실행 노드 및 홉)에 자동화 메시 포트를 추가하여 자동화 메시 기능을 활성화합니다. 모든 노드의 메시 네트워크에 사용되는 기본 포트는 **27199/tcp** 입니다. 인벤토리 파일 내의 각 노드의 변수로 **recptor_listener_port** 를 지정하여 다른 포트를 사용하도록 메시 네트워크를 구성할 수 있습니다.

홉 및 실행 노드 내에서 설치에 사용할 **firewalld** 포트를 설정합니다.

1. **firewalld** 가 실행 중인지 확인합니다.

```
$ sudo systemctl status firewalld
```

2. 컨트롤러 데이터베이스 노드에 **firewalld** 포트를 추가합니다(예: 포트 27199).

```
$ sudo firewall-cmd --permanent --zone=public --add-port=27199/tcp
```

3. **firewalld** 다시 로드:

```
$ sudo firewall-cmd --reload
```

4. 포트가 열려 있는지 확인합니다.

```
$ sudo firewall-cmd --list-ports
```

4.2. ANSIBLE AUTOMATION PLATFORM 인스턴스를 백업

현재 환경의 콘텐츠 및 구성을 저장하는 **backup_dir** 플래그와 함께 **.setup.sh** 스크립트를 실행하여 기존 Ansible Automation Platform 인스턴스를 백업합니다.

1. **ansible-tower-setup-latest** 디렉터리로 이동합니다.

2. 아래 예에 따라 **./setup.sh** 스크립트를 실행합니다.

```
$ ./setup.sh -e 'backup_dir=/ansible/mybackup' -e 'use_compression=True' @credentials.yml
-b (1) (2)
```

(1) **backup_dir** 은 백업을 저장할 디렉터리를 지정합니다.

2 **@credentials.yml** 은 password 변수와 **ansible-vault** 를 통해 암호화된 값을 전달합니다.

백업에 성공하면 **/ansible/mybackup/tower-backup-latest.tar.gz** 에 백업 파일이 생성됩니다.

이 백업은 나중에 이전 인스턴스에서 새 인스턴스로 콘텐츠를 마이그레이션하는데 필요합니다.

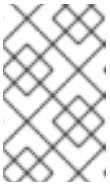
4.3. 병렬 업그레이드를 위해 새 인스턴스 배포

side-by-side 업그레이드 프로세스를 진행하려면 동일한 인스턴스 그룹 구성으로 Ansible Tower 3.8.x의 두 번째 인스턴스를 배포합니다. 이 새 인스턴스는 원래 인스턴스의 콘텐츠 및 구성을 수신하며 나중에 Red Hat Ansible Automation Platform 2.4로 업그레이드됩니다.

4.3.1. Ansible Tower의 새 인스턴스 배포

새 Ansible Tower 인스턴스를 배포하려면 다음을 수행합니다.

1. Ansible Tower 설치 프로그램 페이지로 이동하여 원래 Tower 인스턴스와 일치하는 Tower 설치 프로그램 버전을 다운로드합니다.
2. 설치 프로그램으로 이동한 다음 텍스트 편집기에서 **인벤토리** 파일을 열어 Tower 설치에 대한 **인벤토리** 파일을 구성합니다.
   a. Tower 구성 외에도 **isolated_group** 또는 **instance_group** 이 포함된 필드를 제거합니다.



참고

Ansible Automation Platform 설치 프로그램을 사용하여 Tower를 설치하는 방법에 대한 자세한 내용은 특정 설치 시나리오에 대한 Ansible Automation Platform 설치 가이드를 참조하십시오.

3. **setup.sh** 스크립트를 실행하여 설치를 시작합니다.

새 인스턴스가 설치되면 원래 Tower 인스턴스의 인스턴스 그룹과 일치하도록 Tower 설정을 구성합니다.

4.3.2. 새 인스턴스에서 인스턴스 그룹을 다시 생성

새 인스턴스에서 인스턴스 그룹을 다시 생성하려면 다음을 수행합니다.

참고

원래 Tower 인스턴스의 모든 인스턴스 그룹을 기록해 둡니다. 이러한 그룹을 새 인스턴스에서 다시 생성해야 합니다.

1. Tower의 새 인스턴스에 로그인합니다.
2. 탐색 창에서 **관리 → 인스턴스 그룹을** 선택합니다.
3. **인스턴스 그룹 만들기를** 클릭합니다.
4. 원래 인스턴스의 인스턴스 그룹과 일치하는 이름을 입력한 다음 **저장**을 클릭합니다.
5. **원래 인스턴스의 모든 인스턴스 그룹이 다시 생성될 때까지 반복합니다.**

4.4. 새 인스턴스로 백업 복원

restore_backup_file 플래그를 사용하여 **./setup.sh** 스크립트를 실행하면 원래 1.x 인스턴스의 백업 파일에서 새 인스턴스로 콘텐츠가 마이그레이션됩니다. 이렇게 하면 모든 작업 기록, 템플릿 및 기타 Ansible Automation Platform 관련 콘텐츠가 효과적으로 마이그레이션됩니다.

절차

1. 다음 명령을 실행합니다.

```
$ ./setup.sh -r -e 'restore_backup_file=/ansible/mybackup/tower-backup-latest.tar.gz' -e
'use_compression=True' -e @credentials.yml -r -- --ask-vault-pass 1 2 3
```

1. **restore_backup_file** 은 Ansible Automation Platform 백업 데이터베이스의 위치를 지정합니다.
2. **Use_compression** 은 백업 프로세스 중에 압축이 사용되기 때문에**True** 로 설정됩니다.
3. **-** R은 복원 데이터베이스 옵션을**True**로 설정합니다.

2. 새로운 RHEL 8 Tower 3.8 인스턴스에 로그인하여 원래 인스턴스의 콘텐츠가 복원되었는지 확인합니다.
   a. Administration → Instance Groups (인스턴스 그룹) 로 이동합니다. 이제 다시 생성된 인스턴스 그룹에 원래 인스턴스의 합계 작업이 포함되어야 합니다.
   b. 측면 탐색 패널을 사용하여 작업, 템플릿, iPXE, 자격 증명 및 사용자를 포함하여 원래 인스턴스에서 콘텐츠를 가져왔는지 확인합니다.

이제 원래 인스턴스의 모든 Ansible 콘텐츠가 포함된 Ansible Tower의 새 인스턴스가 있습니다.

이 새 인스턴스를 Ansible Automation Platform 2.4로 업그레이드하여 원래 인스턴스를 덮어쓰지 않고 이전 데이터를 모두 유지합니다.

4.5. ANSIBLE AUTOMATION PLATFORM 2.4로 업그레이드

Ansible Tower의 인스턴스를 Ansible Automation Platform 2.4로 업그레이드하려면 원래 Tower 인스턴스에서 새 Tower 인스턴스로 인벤토리 파일을 복사하고 설치 프로그램을 실행합니다. Red Hat Ansible Automation Platform 설치 프로그램은 사전-2.4를 감지하고 업그레이드 프로세스를 계속하기 위해 업그레이드된 인벤토리 파일을 제공합니다.

1. Red Hat Ansible Automation Platform 다운로드 페이지에서 Red Hat Ansible Automation Platform의 최신 설치 프로그램을 다운로드합니다.
2. 파일을 추출합니다.

```
$ tar xvzf ansible-automation-platform-setup-<latest_version>.tar.gz
```

3. Ansible Automation Platform 설치 디렉터리로 이동합니다.

```
$ cd ansible-automation-platform-setup-<latest_version>/
```

4. 원래 인스턴스의 인벤토리 파일을 최신 설치 프로그램의 디렉터리에 복사합니다.

```
$ cp ansible-tower-setup-3.8.x.x/inventory ansible-automation-platform-
setup-<latest_version>
```

5. **setup.sh** 스크립트를 실행합니다.

```
$ ./setup.sh
```

설정 스크립트를 일시 중지하고 "pre-2.x" 인벤토리 파일이 감지되었지만 원래 인스턴스를 계속 업그레이드할 수 있도록 **inventory.new.ini** 라는 새 파일을 제공합니다.

6. 텍스트 편집기를 사용하여 **inventory.new.ini** 를 엽니다.

참고

설치 스크립트를 실행하여 설치 관리자는 원래 인벤토리 파일에서 [automationcontroller]로 이름 변경과 같은 몇 가지 필드를 수정했습니다.

7. 관련 변수, 노드 및 관련 노드-노드 피어 연결을 할당하여 자동화 메시를 구성하도록 새로 생성된 **inventory.new.ini** 파일을 업데이트합니다.

참고

자동화 메시 토폴로지 설계는 환경의 자동화 요구 사항에 따라 다릅니다. 가능한 모든 시나리오에 대한 설계를 제공하기 위해 이 문서의 범위를 벗어납니다. 다음은 자동화 메시 설계의 예입니다.

홉 노드를 활용하는 세 개의 노드로 구성된 표준 컨트롤 플레인이 있는 인벤토리 파일의 예:

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
control-plane-3.example.com

[automationcontroller:vars]
node_type=control (1)
peers=execution_nodes (2)


[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com peers=execution-node-3.example.com
execution-node-3.example.com peers=execution-node-4.example.com
execution-node-4.example.com peers=execution-node-5.example.com node_type=hop
execution-node-5.example.com peers=execution-node-6.example.com node_type=hop (3)
execution-node-6.example.com peers=execution-node-7.example.com
execution-node-7.example.com

[execution_nodes:vars]
node_type=execution
```

(1) 일반 작업이 아닌 프로젝트 및 인벤토리 업데이트 및 시스템 작업을 실행하는 제어 노드를 지정합니다. 이러한 노드에서 실행 기능이 비활성화됩니다.

2 **[execution_nodes]** 그룹에서 노드 간 연결에 대한 피어 관계를 지정합니다.

3 트래픽을 다른 실행 노드로 라우팅하는 홉 노드를 지정합니다. 홉 노드는 자동화를 실행할 수 없습니다.

8. 자동화 허브 API 토큰을 가져오거나 생성합니다.
   - **automationhub_api_token** 플래그를 사용하여 기존 API 토큰을 가져옵니다.

     ```
     automationhub_api_token=<api_token>
     ```

   - **generate_automationhub_token** 플래그를 **True:**로 설정하여 새 API 토큰을 생성하고 기존 토큰을 무효화합니다.

     ```
     generate_automationhub_token=True
     ```

9. 자동화 메시를 위한 **inventory.new.ini** 구성을 완료한 후 **inventory.new.ini** 를 사용하여 설정 스크립트를 실행합니다.

   ```
   $ ./setup.sh -i inventory.new.ini -e @credentials.yml -- --ask-vault-pass
   ```

10. 설치가 완료되면 모든 자동화 컨트롤러 노드에서 Ansible Automation Platform 대시보드 UI에 로그인하여 Ansible Automation Platform이 성공적으로 설치되었는지 확인합니다.

추가 리소스

- Ansible Automation Platform 설치 프로그램 사용에 대한 일반적인 정보는 Red Hat Ansible Automation Platform 설치 가이드를 참조하십시오.

4.6. 업그레이드된 ANSIBLE AUTOMATION PLATFORM 구성

4.6.1. 자동화 컨트롤러 인스턴스 그룹 구성

Red Hat Ansible Automation Platform 인스턴스를 업그레이드한 후 자동화 컨트롤러 UI에서 설정을 구성하여 원래 인스턴스를 해당 인스턴스 그룹에 연결합니다.

1. 새 컨트롤러 인스턴스에 로그인합니다.
2. 인증 정보, 작업, 인벤토리와 같은 이전 인스턴스의 콘텐츠가 이제 컨트롤러 인스턴스에 표시되어야 합니다.
3. Administration → Instance Groups (인스턴스 그룹) 로 이동합니다.
4. 인스턴스 그룹을 클릭하여 실행 노드를 연결한 다음 Instances 탭을 클릭합니다.
5. **Associate** 를 클릭합니다. 이 인스턴스 그룹에 연결할 노드를 선택한 다음**저장** 을 클릭합니다.
6. 새 실행 노드의 연결을 끊으도록 기본 인스턴스를 수정할 수도 있습니다.

5장. ANSIBLE 콘텐츠 마이그레이션

ansible-core 버전에서 **ansible-core** 2.13으로 마이그레이션하는 경우Ansible 코어 이식 가이드를 검토하여 각 버전 간 변경 및 업데이트를 숙지하십시오. Ansible 코어 포트 지정 가이드를 검토할 때 가이드의 왼쪽 상단 열에 있는 **ansible-core** 또는 **devel** 의 최신 버전을 선택해야 합니다.

완전히 지원 및 인증된 Ansible 콘텐츠 컬렉션 목록은 console.redhat.com 에서 Ansible Automation hub 를 참조하십시오.

5.1. ANSIBLE 컬렉션 설치

이전 Ansible 버전에서 최신 버전으로의 마이그레이션의 일환으로 사용 중인 모듈이 포함된 컬렉션을 찾아 다운로드해야 합니다. 해당 컬렉션 목록을 찾으면 다음 옵션 중 하나를 사용하여 컬렉션을 로컬로 포함할 수 있습니다.

1. **ansible-builder** 를 사용하여 컬렉션을 다운로드하여 런타임 또는 실행 환경에 설치합니다.
2. Automation Controller 프로젝트에서 역할 및 컬렉션을 설치하는 'requirements.yml' 파일을 업데이트합니다. 이렇게 하면 자동화 컨트롤러에서 프로젝트를 동기화할 때마다 역할과 컬렉션이 다운로드됩니다.

참고

대부분의 경우 업스트림 및 다운스트림 컬렉션은 동일할 수 있지만 항상 Automation Hub에서 인증된 컬렉션을 다운로드합니다.

5.2. ANSIBLE 플레이북 및 역할을 CORE 2.13으로 마이그레이션

컬렉션 기반 콘텐츠에서 컬렉션 기반 콘텐츠로 마이그레이션하는 경우 예기치 않은 동작을 방지하려면 플레이북 및 역할에 FQCN(완전한 컬렉션 이름)을 사용해야 합니다.

FQCN이 있는 플레이북의 예:

```
- name: get some info
  amazon.aws.ec2_vpc_net_info:
    region: "{{ec2_region}}"
  register: all_the_info
  delegate_to: localhost
  run_once: true
```

ansible-core 모듈을 사용하고 다른 컬렉션에서 모듈을 호출하지 않는 경우 FQCN **ansible.builtin.copy** 를 사용해야 합니다.

FQCN이 있는 모듈의 예:

```
- name: copy file with owner and permissions
  ansible.builtin.copy:
  src: /srv/myfiles/foo.conf
  dest: /etc/foo.conf
  owner: foo
  group: foo
  mode: '0644'
```

5.3. 플레이북 예제 변환

예

이 예제는 작업을 실행하는 동안 파일을 읽고 쓸 수 있도록 /mydata 라는 공유 디렉터리입니다. 자동화 실행에 사용할 실행 노드에 이미 있어야 합니다.

기본 호스트에 이미 이 작업이 있으므로 *aape1.local* 실행 노드를 대상으로 합니다.

```
[awx@aape1 ~]$ ls -la /mydata/
total 4
drwxr-xr-x.  2 awx  awx   41 Apr 28 09:27 .
dr-xr-xr-x. 19 root root 258 Apr 11 15:16 ..
-rw-r--r--.  1 awx  awx   33 Apr 11 12:34 file_read
-rw-r--r--.  1 awx  awx    0 Apr 28 09:27 file_write
```

간단한 플레이북을 사용하여 절전으로 자동화를 시작하여 사용자가 액세스할 수 있도록 하고 프로세스를 이해하고 파일에 대한 읽기 및 쓰기를 시연합니다.

```
# vim:ft=ansible:
```

```
- hosts: all
  gather_facts: false
  ignore_errors: yes
  vars:
    period: 120
    myfile: /mydata/file
  tasks:
    - name: Collect only selected facts
      ansible.builtin.setup:
        filter:
          - 'ansible_distribution'
          - 'ansible_machine_id'
          - 'ansible_memtotal_mb'
          - 'ansible_memfree_mb'
    - name: "I'm feeling real sleepy..."
      ansible.builtin.wait_for:
        timeout: "{{ period }}"
      delegate_to: localhost
    - ansible.builtin.debug:
        msg: "Isolated paths mounted into execution node: {{ AWX_ISOLATIONS_PATHS }}"
    - name: "Read pre-existing file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_read'
    - name: "Write to a new file..."
      ansible.builtin.copy:
        dest: "{{ myfile }}_write"
        content: |
          This is the file I've just written to.

    - name: "Read written out file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_write') }}"
```

Ansible Automation Platform 2 탐색 패널에서 설정을 선택합니다. 그런 다음 작업 옵션에서 Job settings 를 선택합니다.

격리된 작업을 노출하는 경로:

```
[
"/mydata:/mydata:rw"
]
```

볼륨 마운트는 컨테이너의 동일한 이름으로 매핑되며 읽기-쓰기 기능이 있습니다. 그러면 작업 템플릿을 시작할 때 사용됩니다.

각 실행의 수면 기간을 조정할 수 있도록 *시작 시 프롬프트*를 *extra_vars* 로 설정해야 합니다. 기본값은 30 초입니다.

시작된 후 *wait_for* 모듈은 sleep에 대해 호출되므로 실행 노드로 이동하여 실행 중인 항목을 확인할 수 있습니다.

실행이 성공적으로 완료되었는지 확인하려면 다음 명령을 실행하여 작업의 출력을 가져옵니다.

```
$ podman exec -it 'podman ps -q' /bin/bash
bash-4.4#
```

이제 실행 중인 실행 환경 컨테이너 내부에 있습니다.

권한을 보면 awx 가 'root'가되었지만 rootless Podman을 사용하고 있기 때문에 사용자를 샌드박스와 유사한 커널 네임스페이스로 매핑하기 때문에 슈퍼유저에서와 같이 루트가 아닙니다. shadow-utils에서 rootless Podman의 작동 방식에 대해 자세히 알아보십시오.

```
bash-4.4# ls -la /mydata/
Total 4
drwxr-xr-x. 2 root root 41 Apr 28 09:27 .
dr-xr-xr-x. 1 root root 77 Apr 28 09:40 ..
-rw-r—--r–. 1 root root 33 Apr 11 12:34 file_read
-rw-r—--r–. 1 root root  0 Apr 28 09:27 file_write
```

결과에 따르면 이 작업이 실패했습니다. 이유를 이해하려면 나머지 출력을 확인해야 합니다.

```
TASK [Read pre-existing file…]****************************** 10:50:12
ok: [localhost] =>  {
    “Msg”: “This is the file I am reading in.”

TASK {Write to a new file...}********************************* 10:50:12
An exception occurred during task execution. To see the full traceback, use -vvv. The error was:
PermissionError: [Errno 13] Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'
/mydata/file_write'
Fatal: [localhost]: FAILED! => {"changed": false, :checksum":
"9f576o85d584287a3516ee8b3385cc6f69bf9ce", "msg": "Unable to make
b'/root/.ansible/tmp/anisible-tim-1651139412.9808054-40-91081834383738/source' into
/mydata/file_write, failed final rename from b'/mydata/.ansible_tmpazyqyqdrfile_write': [Errno 13]
Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'/mydata/file_write}
...ignoring

TASK [Read written out file...] ***************************** 10:50:13
Fatal: [localhost]: FAILED: => {"msg": "An unhandled exception occurred while running the lookup
```

```
plugin 'file'. Error was a <class 'ansible.errors.AnsibleError;>, original message: could not locate file in
lookup: /mydate/file_write. Vould not locate file in lookup: /mydate/file_write"}
...ignoring
```

:rw 가 설정되어 있어도 작업이 실패했으므로 쓰기 기능이 있어야 합니다. 프로세스에서 기존 파일을 읽을 수 있었지만 쓸 수는 없었습니다. 이는 SELinux 보호로 인해 컨테이너에 마운트된 볼륨 콘텐츠에 적절한 라벨을 배치해야 하기 때문입니다. 레이블이 없으면 SELinux에서 프로세스가 컨테이너 내에서 실행되지 않을 수 있습니다. OS에서 설정한 레이블은 Podman에서 변경하지 않습니다. 자세한 내용은 Podman 설명서를 참조하십시오.

이는 일반적인 잘못 해석이 될 수 있습니다. Podman이 공유 볼륨에서 파일 오브젝트의 레이블을 다시 지정하도록 하는 기본값을 :z 로 설정했습니다.

따라서 :z 를 추가하거나 그대로 둘 수 있습니다.

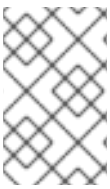
격리된 작업을 노출하는 경로:

```
[
  "/mydata:/mydata"
]
```

이제 플레이북이 예상대로 작동합니다.

```
PLAY [all] ********************************************************* 11:05:52
TASK [I'm feeling real sleepy. . .] *************************** 11:05:52
ok: [localhost]
TASK [Read pre-existing file...] ****************************** 11:05:57
ok: [localhost] => {
 "Msg": "This is the file I'm reading in."
}
TASK [Write to a new file…] ********************************** 11:05:57
ok: [localhost]
TASK [Read written out file…] ******************************** 11:05:58
ok: [localhost] => {
     “Msg”: “This is the file I’ve just written to.”
```

기본 실행 노드 호스트에서 다시, 새로 작성된 콘텐츠가 있습니다.



참고

컨테이너 그룹을 사용하여 Red Hat OpenShift 내부에서 자동화 작업을 시작하는 경우 Ansible Automation Platform 2에 해당 환경에 동일한 경로를 노출하도록 지시할 수 있지만 기본값을 On 으로 전환해야 합니다.

활성화하면 이 값을 *volumeMounts* 및 *실행에* 사용할 Pod 사양 내에 삽입합니다. 다음과 같이 표시됩니다.

```
apiVersion: v1
kind: Pod
Spec:
  containers:
  - image: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8
 args:
   - ansible runner
   - worker
```

```
    - –private-data-dir=/runner
  volumeMounts:
mountPath: /mnt2
name: volume-0
readOnly: true
mountPath: /mnt3
name: volume-1
readOnly: true
mountPath: /mnt4
name: volume-2
readOnly: true
volumes:
hostPath:
  path: /mnt2
  type: “”
name: volume-0
hostPath:
  path: /mnt3
  type: “”
name: volume-1
hostPath:
  path: /mnt4
  type: “”
name: volume-2
```

실행 중인 컨테이너 내부의 스토리지는 오버레이 파일 시스템을 사용하고 있습니다. 실행 중인 컨테이너 내부의 모든 수정 사항은 마운트 해제되는 tmpfs와 유사하게 작업이 완료된 후 삭제됩니다.

6장. AAP2용 플레이북 변환

Ansible Automation Platform 2 및 컨테이너화된 실행 환경을 사용하면 *localhost* 의 사용이 변경되었습니다. 이전 버전의 Ansible Automation Platform에서는 작업이 기본 자동화 컨트롤러 호스트에서 실행 중으로 변환된 *localhost* 에 대해 실행되었습니다. 이는 데이터 및 영구 아티팩트를 저장하는 데 사용할 수 있습니다.

Ansible Automation Platform 2를 사용하면 localhost가 컨테이너 내에서 실행 중임을 의미하며, 이는 본질적으로 임시적입니다. *localhost* 는 더 이상 특정 호스트에 연결되지 않으며 이식 가능한 실행 환경에서 이미 실행 환경 및 소프트웨어 사전 요구 사항이 포함된 올바른 환경 및 소프트웨어 사전 요구 사항으로 실행할 수 있습니다.

6.1. 자동 실행에서 데이터 유지

데이터가 해당 호스트에 연결되므로 로컬 자동화 컨트롤러 파일 시스템을 비생산적인 것으로 간주합니다. 다중 노드 클러스터가 있는 경우 매번 다른 호스트에 연결하여 서로 다른 워크플로우를 생성하는 경우 문제가 발생할 수 있습니다. 예를 들어 다른 노드가 플레이북을 실행하는 동안 한 노드에서만 디렉터리가 생성된 경우 결과가 일치하지 않습니다.

솔루션은 Amazon S3, Gist 또는 데이터를 데이터 끝점에 rsync 역할과 같은 공유 스토리지 솔루션의 일부 형식을 사용하는 것입니다.

옵션은 런타임 시 컨테이너에 데이터 또는 구성을 삽입하는 데 있습니다. 이를 위해 자동화 컨트롤러의 분리된 작업 경로 옵션을 사용할 수 있습니다.

이를 통해 런타임 시 디렉터리 및 파일을 실행 환경에 마운트할 수 있습니다. 이 작업은 ansible-runner를 사용하여 Podman 컨테이너에 삽입하여 자동화를 시작하는 자동화 메시를 통해 수행됩니다. 다음은 격리된 작업 경로를 사용하는 몇 가지 사용 사례입니다.

- 실행 환경으로 배치하는 대신 런타임에 SSL 인증서를 제공합니다.
- SSH 구성 설정과 같은 런타임 구성 데이터를 전달하지만 자동화 중에 사용하려는 모든 항목이 될 수 있습니다.
- 자동화 실행 전, 도중 및 후에 사용된 파일 읽기 및 쓰기.

사용할 수 있는 경고는 다음과 같습니다.

- 볼륨 마운트는 자동화할 수 있는 모든 노드(하이브리드 컨트롤 플레인 노드 및 모든 실행 노드)에서 미리 표시되어야 합니다.
- SELinux가 활성화되어 있는 경우 (Ansible Automation Platform 기본값) 파일 권한에 주의하십시오.
    - rootless Podman은 OCP 기반이 아닌 설치에서 실행되기 때문에 중요합니다.

경고는 신중하게 관찰해야합니다. Ansible Automation Platform 2 내에서 사용되는 작업 경로의 [:OPTIONS] 부분인 rootless Podman 및 Podman 볼륨 마운트 런타임 옵션을 읽는 것이 좋습니다.

추가 리소스

- 루트리스 Podman 이해
- podman 볼륨 마운트 런타임 옵션.

6.1.1. 플레이북 예제 변환

예

이 예제는 작업을 실행하는 동안 파일을 읽고 쓸 수 있도록 /mydata 라는 공유 디렉터리입니다. 자동화 실행에 사용할 실행 노드에 이미 있어야 합니다.

기본 호스트에 이미 이 작업이 있으므로 *aape1.local* 실행 노드를 대상으로 합니다.

```
[awx@aape1 ~]$ ls -la /mydata/
total 4
drwxr-xr-x.  2 awx  awx   41 Apr 28 09:27 .
dr-xr-xr-x. 19 root root 258 Apr 11 15:16 ..
-rw-r--r--.  1 awx  awx   33 Apr 11 12:34 file_read
-rw-r--r--.  1 awx  awx    0 Apr 28 09:27 file_write
```

간단한 플레이북을 사용하여 절전으로 자동화를 시작하여 사용자가 액세스할 수 있도록 하고 프로세스를 이해하고 파일에 대한 읽기 및 쓰기를 시연합니다.

```
# vim:ft=ansible:
```

```
- hosts: all
  gather_facts: false
  ignore_errors: yes
  vars:
    period: 120
    myfile: /mydata/file
  tasks:
    - name: Collect only selected facts
      ansible.builtin.setup:
        filter:
          - 'ansible_distribution'
          - 'ansible_machine_id'
          - 'ansible_memtotal_mb'
          - 'ansible_memfree_mb'
    - name: "I'm feeling real sleepy..."
      ansible.builtin.wait_for:
        timeout: "{{ period }}"
      delegate_to: localhost
    - ansible.builtin.debug:
        msg: "Isolated paths mounted into execution node: {{ AWX_ISOLATIONS_PATHS }}"
    - name: "Read pre-existing file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_read'
    - name: "Write to a new file..."
      ansible.builtin.copy:
        dest: "{{ myfile }}_write"
        content: |
          This is the file I've just written to.

    - name: "Read written out file..."
      ansible.builtin.debug:
        msg: "{{ lookup('file', '{{ myfile }}_write') }}"
```

Ansible Automation Platform 2 탐색 패널에서 설정을 선택합니다. 그런 다음 작업 옵션에서 Job settings 를 선택합니다.

격리된 작업을 노출하는 경로:

```
[
"/mydata:/mydata:rw"
]
```

볼륨 마운트는 컨테이너의 동일한 이름으로 매핑되며 읽기-쓰기 기능이 있습니다. 그러면 작업 템플릿을 시작할 때 사용됩니다.

각 실행의 수면 기간을 조정할 수 있도록 *시작 시 프롬프트*를 *extra_vars* 로 설정해야 합니다. 기본값은 30초입니다.

시작된 후 *wait_for* 모듈은 sleep에 대해 호출되므로 실행 노드로 이동하여 실행 중인 항목을 확인할 수 있습니다.

실행이 성공적으로 완료되었는지 확인하려면 다음 명령을 실행하여 작업의 출력을 가져옵니다.

```
$ podman exec -it 'podman ps -q' /bin/bash
bash-4.4#
```

이제 실행 중인 실행 환경 컨테이너 내부에 있습니다.

권한을 보면 awx 가 'root'가되었지만 rootless Podman을 사용하고 있기 때문에 사용자를 샌드박스와 유사한 커널 네임스페이스로 매핑하기 때문에 슈퍼유저에서와 같이 루트가 아닙니다. shadow-utils에서 rootless Podman의 작동 방식에 대해 자세히 알아보십시오.

```
bash-4.4# ls -la /mydata/
Total 4
drwxr-xr-x. 2 root root 41 Apr 28 09:27 .
dr-xr-xr-x. 1 root root 77 Apr 28 09:40 ..
-rw-r—--r–. 1 root root 33 Apr 11 12:34 file_read
-rw-r—--r–. 1 root root  0 Apr 28 09:27 file_write
```

결과에 따르면 이 작업이 실패했습니다. 이유를 이해하려면 나머지 출력을 확인해야 합니다.

```
TASK [Read pre-existing file…]****************************** 10:50:12
ok: [localhost] =>  {
    “Msg”: “This is the file I am reading in.”

TASK {Write to a new file...}********************************* 10:50:12
An exception occurred during task execution. To see the full traceback, use -vvv. The error was:
PermissionError: [Errno 13] Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'
/mydata/file_write'
Fatal: [localhost]: FAILED! => {"changed": false, :checksum":
"9f576o85d584287a3516ee8b3385cc6f69bf9ce", "msg": "Unable to make
b'/root/.ansible/tmp/anisible-tim-1651139412.9808054-40-91081834383738/source' into
/mydata/file_write, failed final rename from b'/mydata/.ansible_tmpazyqyqdrfile_write': [Errno 13]
Permission denied: b'/mydata/.ansible_tmpazyqyqdrfile_write' -> b'/mydata/file_write}
...ignoring

TASK [Read written out file...] ****************************** 10:50:13
Fatal: [localhost]: FAILED: => {"msg": "An unhandled exception occurred while running the lookup
```

```
plugin 'file'. Error was a <class 'ansible.errors.AnsibleError;>, original message: could not locate file in
lookup: /mydate/file_write. Vould not locate file in lookup: /mydate/file_write"}
...ignoring
```

:rw 가 설정되어 있어도 작업이 실패했으므로 쓰기 기능이 있어야 합니다. 프로세스에서 기존 파일을 읽을 수 있었지만 쓸 수는 없었습니다. 이는 SELinux 보호로 인해 컨테이너에 마운트된 볼륨 콘텐츠에 적절한 라벨을 배치해야 하기 때문입니다. 레이블이 없으면 SELinux에서 프로세스가 컨테이너 내에서 실행되지 않을 수 있습니다. OS에서 설정한 레이블은 Podman에서 변경하지 않습니다. 자세한 내용은 Podman 설명서를 참조하십시오.

이는 일반적인 잘못 해석이 될 수 있습니다. Podman이 공유 볼륨에서 파일 오브젝트의 레이블을 다시 지정하도록 하는 기본값을 :z 로 설정했습니다.

따라서 :z 를 추가하거나 그대로 둘 수 있습니다.

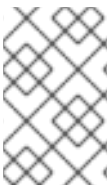
격리된 작업을 노출하는 경로:

```
[
  "/mydata:/mydata"
]
```

이제 플레이북이 예상대로 작동합니다.

```
PLAY [all] ***************************************************** 11:05:52
TASK [I'm feeling real sleepy. . .] *************************** 11:05:52
ok: [localhost]
TASK [Read pre-existing file...] ****************************** 11:05:57
ok: [localhost] => {
 "Msg": "This is the file I'm reading in."
}
TASK [Write to a new file…] ********************************** 11:05:57
ok: [localhost]
TASK [Read written out file…] ******************************** 11:05:58
ok: [localhost] => {
     “Msg”: “This is the file I’ve just written to.”
```

기본 실행 노드 호스트에서 다시, 새로 작성된 콘텐츠가 있습니다.



참고

컨테이너 그룹을 사용하여 Red Hat OpenShift 내부에서 자동화 작업을 시작하는 경우 Ansible Automation Platform 2에 해당 환경에 동일한 경로를 노출하도록 지시할 수 있지만 기본값을 On 으로 전환해야 합니다.

활성화하면 이 값을 *volumeMounts* 및 *실행에* 사용할 Pod 사양 내에 삽입합니다. 다음과 같이 표시됩니다.

```
apiVersion: v1
kind: Pod
Spec:
  containers:
  - image: registry.redhat.io/ansible-automation-platform-24/ee-minimal-rhel8
 args:
   - ansible runner
   - worker
```

```
    - –private-data-dir=/runner
  volumeMounts:
mountPath: /mnt2
name: volume-0
readOnly: true
mountPath: /mnt3
name: volume-1
readOnly: true
mountPath: /mnt4
name: volume-2
readOnly: true
volumes:
hostPath:
  path: /mnt2
  type: “”
name: volume-0
hostPath:
  path: /mnt3
  type: “”
name: volume-1
hostPath:
  path: /mnt4
  type: “”
name: volume-2
```

실행 중인 컨테이너 내부의 스토리지는 오버레이 파일 시스템을 사용하고 있습니다. 실행 중인 컨테이너 내부의 모든 수정 사항은 마운트 해제되는 tmpfs와 유사하게 작업이 완료된 후 삭제됩니다.