Red Hat Ansible Automation Platform 2.4

Red Hat Ansible Automation Platform 설치 가이드

Ansible Automation Platform 설치

Last Updated: 2024-06-13

Red Hat Ansible Automation Platform 2.4 Red Hat Ansible Automation Platform 설치 가이드

Ansible Automation Platform 설치

법적 공지

초록

이 가이드에서는 지원되는 설치 시나리오를 기반으로 Red Hat Ansible Automation Platform을 설치하는 방법을 보여줍니다.

차례

머리말 3

RED HAT 문서에 관한 피드백 제공 4

1장. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 개요 5
1.1. 사전 요구 사항 5

2장. 시스템 요구 사항 7
2.1. RED HAT ANSIBLE AUTOMATION PLATFORM 시스템 요구 사항 7
2.2. 자동화 컨트롤러 시스템 요구 사항 8
2.3. 자동화 허브 시스템 요구 사항 10
2.4. 이벤트 기반 ANSIBLE 컨트롤러 시스템 요구 사항 12
2.5. POSTGRESQL 요구 사항 13

3장. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 19
3.1. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 인벤토리 파일 편집 19
3.2. 설치 시나리오에 따른 인벤토리 파일 예 19
3.3. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 설정 스크립트 실행 32
3.4. 자동화 컨트롤러 설치 확인 32
3.5. 자동화 허브 설치 확인 33
3.6. 이벤트 기반 ANSIBLE 컨트롤러 설치 확인 34

4장. 연결이 해제된 설치 35
4.1. 사전 요구 사항 35
4.2. 연결이 끊긴 RHEL에 ANSIBLE AUTOMATION PLATFORM 설치 35
4.3. REPOSYNC를 사용하여 RPM 리포지토리 동기화 35
4.4. 리포지토리를 호스팅할 새 웹 서버 생성 36
4.5. 로컬에 마운트된 DVD에서 RPM 리포지토리에 액세스 37
4.6. 인터넷 연결 없이 ANSIBLE AUTOMATION PLATFORM에 서브스크립션 매니페스트 추가 38
4.7. ANSIBLE AUTOMATION PLATFORM 설치 번들 다운로드 및 설치 39
4.8. 사후 설치 작업 완료 40
4.9. 프라이빗 자동화 허브로 컬렉션 가져오기 42
4.10. 컬렉션 네임스페이스 생성 42
4.11. 가져온 컬렉션 승인 43
4.12. 연결이 끊긴 환경에서 실행 환경 빌드 44
4.13. 마이너 ANSIBLE AUTOMATION PLATFORM 릴리스 간 업그레이드 50

부록 A. 인벤토리 파일 변수 51
A.1. 일반 변수 51
A.2. ANSIBLE 자동화 허브 변수 52
A.3. 자동화 컨트롤러 변수 64
A.4. ANSIBLE 변수 68
A.5. 이벤트 기반 ANSIBLE 컨트롤러 변수 70

머리말

Red Hat Ansible Automation Platform에 관심을 가져 주셔서 감사합니다. Ansible Automation Platform은 Ansible 기반 환경에 제어, 지식 및 위임을 추가하여 팀이 복잡한 다중 계층 배포를 관리하는 데 도움이 되는 상용 서비스입니다.

이 가이드를 통해 Ansible Automation Platform 설치 후의 설치 요구 사항 및 프로세스를 이해하는 데 도움이 됩니다. 이 문서는 Ansible Automation Platform의 최신 릴리스에 대한 정보를 포함하도록 업데이트되었습니다.

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 설치 개요

Red Hat Ansible Automation Platform 설치 프로그램은 유연성을 제공하므로 지원되는 여러 설치 시나리오를 사용하여 Ansible Automation Platform을 설치할 수 있습니다. Ansible Automation Platform 2.4부터 설치 시나리오에는 IT 요청의 자동화된 해결을 도입하는 Event-Driven Ansible 컨트롤러의 선택적 배포가 포함됩니다.

선택한 설치 시나리오에 관계없이 Ansible Automation Platform을 설치하려면 다음 단계가 포함됩니다.

Red Hat Ansible Automation Platform 설치 프로그램 인벤토리 파일 편집

Ansible Automation Platform 설치 프로그램 인벤토리 파일을 사용하면 설치 시나리오를 지정하고 Ansible에 대한 호스트 배포를 설명할 수 있습니다. 이 문서에 제공된 예제에서는 배포에 해당 시나리오를 설치하는 데 필요한 매개변수 사양을 보여줍니다.

Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

설정 스크립트는 인벤토리 파일에 정의된 필수 매개 변수를 사용하여 프라이빗 자동화 허브를 설치합니다.

자동화 컨트롤러 설치 확인

Ansible Automation Platform을 설치한 후 자동화 컨트롤러에 로그인하여 설치에 성공했는지 확인할 수 있습니다.

자동화 허브 설치 확인

Ansible Automation Platform을 설치한 후 자동화 허브에 로그인하여 설치에 성공했는지 확인할 수 있습니다.

이벤트 기반 Ansible 컨트롤러 설치 확인

Ansible Automation Platform을 설치한 후 이벤트 기반 Ansible 컨트롤러에 로그인하여 설치에 성공했는지 확인할 수 있습니다.

추가 리소스

지원되는 설치 시나리오에 대한 자세한 내용은 Red Hat Ansible Automation Platform 계획 가이드를 참조하십시오.

1.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 받을 수 있습니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치하고 있습니다.
- 모든 패키지를 최신 버전의 RHEL 노드로 업데이트했습니다.

주의

오류를 방지하려면 Ansible Automation Platform을 설치하기 전에 RHEL 노드를 완전히 업그레이드하십시오.

- 레지스트리 서비스 계정 생성의 지침을 사용하여 Red Hat 레지스트리 서비스 계정을 생성 했습니다.

추가 리소스

플랫폼 설치 프로그램 또는 시스템 요구 사항을 얻는 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform *계획 가이드의 Red Hat Ansible Automation Platform 시스템 요구 사항을 참조하십시오* .

2장. 시스템 요구 사항

Red Hat Ansible Automation Platform 설치를 계획하고 사용 사례에 맞는 자동화 메시 토폴로지를 설계할 때 이 정보를 사용하십시오.

사전 요구 사항

- **sudo** 명령을 통해 또는 권한 에스컬레이션을 통해 root 액세스 권한을 얻을 수 있습니다. 권한 에스컬레이션에 대한 자세한 내용은 권한 에스컬레이션 이해 를 참조하십시오.
- AWX, PostgreSQL, Event-Driven Ansible 또는 Pulp와 같은 권한을 root에서 사용자로 분리할 수 있습니다.
- 모든 노드에 NTP 클라이언트를 구성했습니다. 자세한 내용은 Chrony를 사용하여 NTP 서버 구성을 참조하십시오.

2.1. RED HAT ANSIBLE AUTOMATION PLATFORM 시스템 요구 사항

Red Hat Ansible Automation Platform을 설치하고 실행하려면 시스템이 다음과 같은 최소 시스템 요구 사항을 충족해야 합니다.

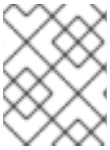
표 2.1. 기본 시스템

요구 사항	필수 항목	참고
서브스크립션	유효한 Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.6 이상 64비트(x86, ppc64le, s390x, aarch64)	Red Hat Ansible Automation Platform은 OpenShift에서 지원됩니다. 자세한 내용은 OpenShift Container Platform에 Red Hat Ansible Automation Platform Operator 배포를 참조하십시오.
ansible-core	ansible-core 버전 2.14 이상	Ansible Automation Platform에는 ansible-core 2.15가 포함된 실행 환경이 포함되어 있습니다.
Python	3.9 이상	
브라우저	현재 지원되는 Mozilla FireFox 또는 Google Chrome 버전	
데이터베이스	PostgreSQL 버전 13	

프로젝트 업데이트 및 컬렉션을 사용하려면 다음이 필요합니다.

- *Table 5.9에 나열된 네트워크 포트 및 프로토콜이 있는지 확인합니다. Automation Hub* 를 사용하여 자동화 허브 또는 Ansible Galaxy 서버에서 컬렉션을 성공적으로 연결하고 다운로드할 수 있습니다.

- 자체 서명된 인증서를 사용하는 경우 또는 Red Hat 도메인을 사용하는 경우 SSL 검사를 비활성화합니다.

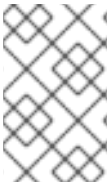


참고

Ansible Automation Platform에서 관리하는 시스템의 요구 사항은 Ansible과 동일합니다. Ansible 커뮤니티 설명서에서 Ansible 설치 단원을 참조하십시오.

Red Hat Ansible Automation Platform 요구 사항에 대한 추가 정보

- Red Hat Ansible Automation Platform은 Ansible 플레이북에 따라 다르며 ansible-core의 안정적인 최신 버전을 설치해야 합니다. ansible-core를 수동으로 다운로드하거나 Red Hat Ansible Automation Platform 설치의 일부로 자동으로 다운로드할 수 있습니다.
- 새로운 설치의 경우 자동화 컨트롤러는 ansible-core의 최신 릴리스 패키지를 설치합니다.
- 번들로 제공되는 Ansible Automation Platform 설치를 수행하는 경우 설치 setup.sh 스크립트에서 번들에서 ansible-core(및 해당 종속 항목)를 설치하려고 합니다.
- Ansible을 수동으로 설치한 경우 Ansible Automation Platform 설치 setup.sh 스크립트에서 Ansible이 설치되었음을 감지하고 다시 설치하지 않습니다.



참고

Red Hat Ansible Automation Platform이 제대로 작동하려면 **dnf** 와 같은 패키지 관리자를 사용하여 Ansible을 설치하고 안정적인 최신 버전의 패키지 관리자를 설치해야 합니다. 버전 2.4 이상에는 Ansible 버전 2.14가 필요합니다.

2.2. 자동화 컨트롤러 시스템 요구 사항

자동화 컨트롤러는 다양한 소프트웨어 구성 요소를 함께 배치하거나 여러 컴퓨팅 노드에 배포할 수 있는 분산 시스템입니다. 설치 프로그램에서 사용 사례에 적합한 토폴로지를 컨트롤, 하이브리드, 실행 및 홉 노드에 적합한 토폴로지를 설계할 수 있도록 네 가지 노드 유형이 추상화로 제공됩니다.

노드 크기 조정에는 다음 권장 사항을 사용하십시오.

참고

제어 및 하이브리드 노드에서 실행 환경 스토리지를 위해 최소 20GB를 **/var/lib/awx** 에 할당합니다.

실행 노드

실행 노드는 자동화를 실행합니다. 더 많은 포크를 실행하는 데 필요한 용량을 늘리려면 메모리 및 CPU를 늘립니다.

참고

- 명시된 RAM 및 CPU 리소스는 실행 노드에 설치된 패키지에 필요하지 않을 수 있지만 평균 작업 수를 동시에 실행하려면 노드의 작업 부하를 처리하는 데 권장되는 최소 수준입니다.
- 권장되는 RAM 및 CPU 노드 크기는 제공되지 않습니다. 필요한 RAM 또는 CPU는 해당 환경에서 실행 중인 작업 수에 따라 직접 달라집니다.

필요한 RAM 및 CPU 수준에 대한 자세한 내용은 자동화 컨트롤러의 성능 튜닝을 참조하십시오.

표 2.2. 실행 노드

요구 사항	최소 요구 사항
RAM	16GB
CPUs	4
로컬 디스크	최소 40GB

컨트롤 노드

컨트롤 노드는 이벤트를 처리하고 프로젝트 업데이트 및 정리 작업을 포함하여 클러스터 작업을 실행합니다. CPU 및 메모리를 늘리면 작업 이벤트 처리에 도움이 될 수 있습니다.

표 2.3. 컨트롤 노드

요구 사항	최소 요구 사항
RAM	16GB
CPUs	4
로컬 디스크	• /var/lib/awx에서 최소 20GB를 사용할 수 있는 최소 40GB • 스토리지 볼륨은 최소 1500 IOPS 기준으로 평가되어야 합니다. • 프로젝트는 제어 및 하이브리드 노드에 저장되며 작업 기간 동안 실행 노드에도 저장됩니다. 클러스터에 많은 대규모 프로젝트가 있는 경우 디스크 공간 오류를 방지하기 위해 /var/lib/awx/projects에서 GB를 두 배로 늘리는 것이 좋습니다.

홉 노드

홉 노드는 자동화 메시의 한 부분에서 다른 부분으로 트래픽을 라우팅하는 역할을 합니다(예: 홉 노드는 다른 네트워크로 홉 호스트가 될 수 있음). RAM은 처리량에 영향을 미칠 수 있으며 CPU 활동이 적습니다. 네트워크 대역폭과 대기 시간은 일반적으로 RAM 또는 CPU보다 더 중요한 요소입니다.

표 2.4. 홉 노드

요구 사항	최소 요구 사항
RAM	16GB
CPUs	4
로컬 디스크	40GB

- 실제 RAM 요구 사항은 동시에 관리할 호스트 자동화 컨트롤러 수(작업 템플릿의 **forks** 매개변수 또는 시스템 **ansible.cfg** 파일에 의해 제어됨)에 따라 달라집니다. 가능한 리소스 충돌을 방지하기 위해 Ansible은 10 포크당 1GB의 메모리와 자동화 컨트롤러에 대해 2GB의 메모리를 권장합니다. 자세한 내용은 자동화 컨트롤러 용량 결정 및 작업 영향을 참조하십시오. **포크를** 400으로 설정하면 42GB의 메모리가 권장됩니다.
- 자동화 컨트롤러 호스트는 Cryostat 가 0022로 설정되어 있는지 확인합니다. 그렇지 않으면 설정이 실패합니다. 이 오류를 방지하려면 **Cryostat=0022** 를 설정합니다.
- 더 많은 수의 호스트를 처리할 수 있지만 포크 번호가 총 호스트 수보다 작으면 호스트 간에 더 많은 패스가 필요합니다. 다음 접근 방법 중 하나를 사용하여 이러한 RAM 제한을 방지할 수 있습니다.
  - 롤링 업데이트를 사용합니다.
  - 자동화 컨트롤러에 빌드된 프로비저닝 콜백 시스템을 사용합니다. 각 시스템은 구성을 요청하는 각 시스템이 대기열로 전환하고 가능한 한 빨리 처리됩니다.
  - 자동화 컨트롤러에서 AMI와 같은 이미지를 생성하거나 배포하는 경우입니다.

추가 리소스

- 자동화 컨트롤러 서브스크립션을 얻는 방법에 대한 자세한 내용은 서브스크립션 가져오기를 참조하십시오.
- 질문이 있는 경우 Red Hat 고객 포털 을 통해 Ansible 지원에 문의하십시오.

2.3. 자동화 허브 시스템 요구 사항

Automation Hub를 사용하면 Red Hat Ansible 및 Certified Partners에서 새로 인증된 자동화 콘텐츠를 검색하고 사용할 수 있습니다. Ansible 자동화 허브에서는 클라우드 자동화, 네트워크 자동화 및 보안 자동화와 같은 사용 사례에 대해 Red Hat 및 파트너가 개발한 자동화 콘텐츠인 Ansible 컬렉션을 검색하고 관리할 수 있습니다.

Automation Hub에는 다음과 같은 시스템 요구 사항이 있습니다.

요구 사항	필수 항목	참고
RAM	최소 8GB	• 8GB RAM(Oplud 평가판 설치 시 최소 및 권장) • 8GB RAM(외부 독립 실행형 PostgreSQL 데이터베이스의 경우 최소) • 구성의 포크를 기반으로 하는 용량은 자동화 컨트롤러 용량 결정 및 작업 영향을 참조하십시오.
CPUs	최소 2개	구성의 포크를 기반으로 하는 용량은 자동화 컨트롤러 용량 결정 및 작업 영향을 참조하십시오.
로컬 디스크	60GB 디스크	컬렉션 저장을 위해 최소 40GB를 **/var** 에 전용으로 지정합니다.

참고

프라이빗 자동화 허브

내부 주소에서 프라이빗 자동화 허브를 설치하고 외부 주소만 포함하는 인증서가 있는 경우 설치 시 인증서 문제 없이 컨테이너 레지스트리로 사용할 수 없습니다.

이를 방지하려면 설치 인벤토리 파일의 프라이빗 자동화 허브 노드에 연결하는 https://pah.example.com와 같은 값과 함께 **automationhub_main_url** 인벤토리 변수를 사용합니다.

그러면 외부 주소가 **/etc/pulp/settings.py** 에 추가됩니다. 즉, 외부 주소만 사용해야 합니다.

인벤토리 파일 변수에 대한 자세한 내용은 *Red Hat Ansible Automation Platform 설치 가이드의 인벤토리 파일 변수를 참조하십시오.*

2.3.1. 고가용성 자동화 허브 요구사항

HA(고가용성) 자동화 허브를 배포하기 전에 환경에 공유 파일 시스템이 설치되어 있고 필요한 경우 네트워크 스토리지 시스템을 구성했는지 확인합니다.

2.3.1.1. 필수 공유 파일 시스템

고가용성 자동화 허브를 사용하려면 NFS와 같은 공유 파일 시스템이 환경에 이미 설치되어 있어야 합니다. Red Hat Ansible Automation Platform 설치 프로그램을 실행하기 전에 클러스터 전체에 **/var/lib/pulp** 디렉터리를 공유 파일 시스템 설치의 일부로 설치했는지 확인합니다. Red Hat Ansible Automation Platform 설치 프로그램은 노드 중 하나에서 **/var/lib/pulp** 이 탐지되지 않는 경우 오류를 반환하여 고가용성 자동화 허브 설정이 실패합니다.

노드 중 하나에서 **/var/lib/pulp** 가 탐지되지 않는 오류가 발생하면 모든 서버에 **/var/lib/pulp** 가 제대로 마운트되고 설치 프로그램을 다시 실행합니다.

2.3.1.2. 네트워크 스토리지를 위한 firewalld 설치

자동화 허브 노드 자체에 네트워크 스토리지를 사용하여 HA 자동화 허브를 설치하려면 먼저 **firewalld** 를 설치하고 사용하여 Ansible Automation Platform 설치 프로그램을 실행하기 전에 공유 스토리지 시스템에 필요한 포트를 열어야 합니다.

다음 명령을 실행하여 **firewalld** 를 설치하고 구성합니다.

1. **firewalld** 데몬을 설치합니다.

```
$ dnf install firewalld
```

2. 다음 명령을 사용하여 <service> 아래에 네트워크 스토리지를 추가합니다.

```
$ firewall-cmd --permanent --add-service=<service>
```

참고

지원되는 서비스 목록은 **$ firewall-cmd --get-services** 명령을 사용하십시오.

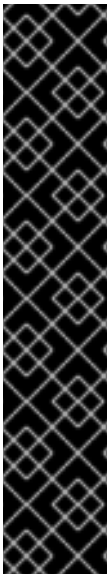
3. 구성을 적용하려면 다시 로드합니다.

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

2.4. 이벤트 기반 ANSIBLE 컨트롤러 시스템 요구 사항

이벤트 기반 Ansible 컨트롤러는 CPU 코어 수에 따라 필요에 따라 다양한 장기 실행 프로세스(예: 룰북 활성화)를 처리할 수 있는 단일 노드 시스템입니다. 다음 최소 요구 사항을 사용하여 기본적으로 최대 12개의 동시 활성화를 실행합니다.

요구 사항	필수 항목
RAM	16GB
CPUs	4
로컬 디스크	최소 40GB



중요

- Red Hat Enterprise Linux 8을 실행 중이고 메모리 제한을 설정하려면 이벤트 기반 Ansible을 설치하기 전에 cgroup v2가 활성화되어 있어야 합니다. 자세한 내용은 Knowledge-Centered Support (KCS) 문서를 참조하십시오. Ansible Automation Platform Event-Driven Ansible Controller for Red Hat Enterprise Linux 8 requires cgroupv2.

- 표준 조건에서 이벤트 기반 Ansible 룰북을 활성화하면 약 250MB의 메모리를 사용합니다. 그러나 실제 메모리 사용은 규칙의 복잡성과 처리된 이벤트의 볼륨 및 크기에 따라 크게 다를 수 있습니다. 많은 수의 이벤트가 예상되거나 룰북 복잡성이 높은 시나리오에서는 스테이징 환경에서 리소스 사용을 사전 평가하십시오. 이렇게 하면 최대 활성화 수가 리소스 용량을 기반으로 합니다. 이벤트 기반 Ansible 컨트롤러 최대 실행 가능한 활성화 설정에 대한 예는 단일 자동화 컨트롤러, 단일 자동화 허브 및 외부(installer managed) 데이터베이스가 포함된 단일 자동화 컨트롤러, 단일 이벤트 기반 Ansible 컨트롤러 노드를 참조하십시오.

2.5. POSTGRESQL 요구 사항

Red Hat Ansible Automation Platform은 PostgreSQL 13을 사용합니다. PostgreSQL 사용자 암호는 데이터베이스에 저장하기 전에 SCRAM-SHA-256 보안 해시 알고리즘으로 해시됩니다.

자동화 컨트롤러 인스턴스가 데이터베이스에 액세스할 수 있는지 확인하려면 명령 **awx-manage check_db** 명령을 사용하여 수행할 수 있습니다.

표 2.5. 데이터베이스

Service	필수 항목	참고

Service	필수 항목	참고
데이터베이스	• 20GB 전용 하드 디스크 공간 • 4개의 CPU • 16GB RAM	• 150GB 이상 권장 • 스토리지 볼륨은 높은 기준 IOPS(1500 이상)로 평가되어야 합니다. • 모든 자동화 컨트롤러 데이터는 데이터베이스에 저장됩니다. 데이터베이스 스토리지는 관리되는 호스트 수, 작업 실행 수, 팩트 캐시에 저장된 팩트 수, 개별 작업의 작업 수에 따라 증가합니다. 예를 들어 플레이북이 250개 호스트에서 매시간(하루 24회) 실행되고 20개의 작업이 포함된 플레이북은 매주 데이터베이스에 800000개 이상의 이벤트를 저장합니다. • 데이터베이스에 공간이 충분하지 않은 경우 이전 작업 실행 및 팩트를 정기적으로 정리해야 합니다. 자세한 내용은 *자동화 컨트롤러 관리 가이드 의 관리 작업을 참조하십시오.*

PostgreSQL 설정

선택적으로 Red Hat Ansible Automation Platform 설치 프로그램에서 관리하지 않는 별도의 노드로 PostgreSQL 데이터베이스를 구성할 수 있습니다. Ansible Automation Platform 설치 프로그램에서 데이터베이스 서버를 관리할 때 대부분의 워크로드에 일반적으로 권장되는 기본값으로 서버를 구성합니다. 데이터베이스 성능을 개선하는 데 사용할 수 있는 설정에 대한 자세한 내용은 데이터베이스 설정을 참조하십시오.

추가 리소스

PostgreSQL 서버 튜닝에 대한 자세한 내용은 PostgreSQL 설명서 를 참조하십시오.

2.5.1. 외부(고객 지원) 데이터베이스 설정

중요

Red Hat은 외부(고객 지원) 데이터베이스를 지원하지 않지만 고객이 사용합니다. 제품 설치 관점에서의 초기 구성에 대한 다음 지침은 관련 지원 요청을 방지하기 위해 제공됩니다.

자동화 컨트롤러와 함께 사용할 외부 PostgreSQL 호환 데이터베이스에서 데이터베이스, 사용자 및 암호를 생성하려면 다음 절차를 사용하십시오.

절차

1. 슈퍼유저 권한으로 PostgreSQL 호환 데이터베이스 서버에 연결합니다.

```
# psql -h <db.example.com> -U superuser -p 5432 -d postgres <Password for user superuser>:
```

다음과 같습니다.

```
-h hostname
--host=hostname
```

서버가 실행 중인 시스템의 호스트 이름을 지정합니다. 값이 슬래시로 시작되면 Unix-domain 소켓의 디렉터리로 사용됩니다.

```
-d dbname
--dbname=dbname
```

연결할 데이터베이스의 이름을 지정합니다. 이는 명령줄에서 첫 번째 비옵션 인수로 **dbname** 을 지정하는 것과 동일합니다. **dbname** 은 연결 문자열일 수 있습니다. 이 경우 연결 문자열 매개 변수는 충돌하는 명령줄 옵션을 재정의합니다.

```
-U username
--username=username
```

기본값 대신 사용자 **이름으로** 데이터베이스에 연결합니다. (이 작업을 수행할 수 있는 권한이 있어야 합니다.)

2. **createDB** 또는 사용자에게 할당된 관리자 역할을 사용하여 사용자, 데이터베이스 및 암호를 만듭니다. 자세한 내용은 데이터베이스 역할을 참조하십시오.

3. 데이터베이스 자격 증명 및 호스트 세부 정보를 자동화 컨트롤러 인벤토리 파일에 외부 데이터베이스로 추가합니다.
다음 예제에서는 기본값을 사용합니다.

```
[database]
pg_host='db.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='redhat'
```

4. 설치 프로그램을 실행합니다.
자동화 컨트롤러가 있는 PostgreSQL 데이터베이스를 사용하는 경우 연결된 사용자가 데이터베이스를 소유하며 **createDB** 또는 관리자 역할이 할당되어야 합니다.

5. 사용자, 암호 및 데이터베이스 이름을 사용하여 생성된 데이터베이스에 연결할 수 있는지 확인합니다.

6. 사용자 권한을 확인합니다. 사용자에게 **createDB** 또는 관리자 역할이 있어야 합니다.

참고

이 절차 중에 외부 데이터베이스 범위를 확인해야 합니다. 자세한 내용은 https://access.redhat.com/articles/4010491에서 참조하십시오.

2.5.2. 자동화 허브 PostgreSQL 데이터베이스의 hstore 확장 활성화

Ansible Automation Platform 2.4에서 데이터베이스 마이그레이션 스크립트는 **hstore** 필드를 사용하여 정보를 저장하므로 자동화 허브 PostgreSQL 데이터베이스에 대한 **hstore** 확장을 활성화해야 합니다.

이 프로세스는 Ansible Automation Platform 설치 프로그램 및 관리형 PostgreSQL 서버를 사용할 때 자동으로 수행됩니다.

PostgreSQL 데이터베이스가 외부인 경우 자동화 허브를 설치하기 전에 수동으로 **hstore** 확장을 자동화 허브 PostreSQL 데이터베이스로 활성화해야 합니다.

자동화 허브를 설치하기 전에 **hstore** 확장 기능을 활성화하지 않으면 데이터베이스 마이그레이션 중에 오류가 발생합니다.

절차

1. PostgreSQL 서버(자동화 허브 데이터베이스)에서 확장을 사용할 수 있는지 확인합니다.

```
$ psql -d <automation hub database> -c "SELECT * FROM pg_available_extensions WHERE name='hstore'"
```

여기서 <**automation hub database>의 기본값** 은 **automationhub** 입니다.

hstore 를 사용할 수 있는 출력 예

```
name  | default_version | installed_version |comment
------+-----------------+-------------------+-----------------------------------------------------
 hstore | 1.7           |                   | data type for storing sets of (key, value) pairs
(1 row)
```

hstore 를 사용할 수 없는 출력 예

```
 name | default_version | installed_version | comment
------+-----------------+-------------------+---------
(0 rows)
```

2. RHEL 기반 서버에서 **hstore** 확장은 **postgresql-contrib** RPM 패키지에 포함되어 있으며 PostgreSQL 서버 RPM 패키지를 설치할 때 자동으로 설치되지 않습니다.
RPM 패키지를 설치하려면 다음 명령을 사용하십시오.

```
dnf install postgresql-contrib
```

3. 다음 명령을 사용하여 자동화 허브 데이터베이스에 **hstore** PostgreSQL 확장을 생성합니다.

```
$ psql -d <automation hub database> -c "CREATE EXTENSION hstore;"
```

결과는 다음과 같습니다.

```
CREATE EXTENSION
```

4. 다음 출력에서 **installed_version** 필드에는 **hstore** 가 활성화되었음을 나타내는 **hstore** 확장이 포함되어 있습니다.

```
name | default_version | installed_version | comment
-----+-----------------+-------------------+-----------------------------------------------------
hstore |    1.7     |      1.7          | data type for storing sets of (key, value) pairs
(1 row)
```

2.5.3. Ansible Automation Platform PostgreSQL 데이터베이스에 대한 스토리지 성능 벤치마킹

FIO(flexible I/O Tester) 툴을 사용하여 최소 Ansible Automation Platform PostgreSQL 데이터베이스 요구 사항을 충족하는지 확인합니다. FIO는 스토리지 시스템의 IOPS 성능을 작성하고 작성하는 데 사용되는 툴입니다.

사전 요구 사항

- F**io**(flexible I/O Tester) 스토리지 성능 벤치마킹 툴을 설치했습니다.
  fio 를 설치하려면 root 사용자로 다음 명령을 실행합니다.

  ```
  # yum -y install fio
  ```

- **fio** 테스트 데이터 로그 파일을 저장할 충분한 디스크 공간이 있습니다.
  절차에 표시된 예제에는 **/tmp** 디렉토리에 최소 60GB의 디스크 공간이 필요합니다.
  - **numjobs** 는 명령으로 실행하는 작업 수를 설정합니다.
  - **size=10G** 는 각 작업에서 생성한 파일 크기를 설정합니다.
- **size** 매개변수 값을 조정했습니다. 이 값을 조정하면 테스트 데이터 양이 줄어듭니다.

절차

1. 임의의 쓰기 테스트를 실행합니다.

   ```
   $ fio --name=write_iops --directory=/tmp --numjobs=3 --size=10G \
   --time_based --runtime=60s --ramp_time=2s --ioengine=libaio --direct=1 \
   --verify=0 --bs=4K --iodepth=64 --rw=randwrite \
   --group_reporting=1 > /tmp/fio_benchmark_write_iops.log \
   2>> /tmp/fio_write_iops_error.log
   ```

2. 임의의 읽기 테스트를 실행합니다.

   ```
   $ fio --name=read_iops --directory=/tmp \
   --numjobs=3 --size=10G --time_based --runtime=60s --ramp_time=2s \
   --ioengine=libaio --direct=1 --verify=0 --bs=4K --iodepth=64 --rw=randread \
   --group_reporting=1 > /tmp/fio_benchmark_read_iops.log \
   2>> /tmp/fio_read_iops_error.log
   ```

3. 결과를 검토합니다.
   벤치마크 명령으로 작성된 로그 파일에서 **iops** 로 시작하는 행을 검색합니다. 이 행은 테스트의 최소, 최대값 및 평균 값을 표시합니다.

   다음 예제에서는 임의의 읽기 테스트에 대한 로그 파일의 행을 보여줍니다.

   ```
   $ cat /tmp/fio_benchmark_read_iops.log
   ```

```
read_iops: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B,
ioengine=libaio, iodepth=64
[…]
  iops        : min=50879, max=61603, avg=56221.33, stdev=679.97, samples=360
[…]
```

자체 비즈니스 요구 사항, 애플리케이션 워크로드 및 새로운 요구에 따라 로그 파일을 검토, 모니터링 및 다시 방문해야 합니다.

3장. RED HAT ANSIBLE AUTOMATION PLATFORM 설치

Ansible Automation Platform은 모듈식 플랫폼입니다. 자동화 허브 및 이벤트 기반 Ansible 컨트롤러와 같은 다른 자동화 플랫폼 구성 요소를 사용하여 자동화 컨트롤러를 배포할 수 있습니다. Ansible Automation Platform과 함께 제공되는 구성 요소에 대한 자세한 내용은 Red Hat Ansible Automation Platform 계획 가이드에서 Red Hat Ansible Automation Platform 구성 요소를 참조하십시오.

Red Hat Ansible Automation Platform에서 지원되는 몇 가지 설치 시나리오가 있습니다. Red Hat Ansible Automation Platform을 설치하려면 인벤토리 파일 매개변수를 편집하여 설치 시나리오를 지정해야 합니다. 다음 중 하나를 자체 인벤토리 파일의 기준으로 사용할 수 있습니다.

- 외부(installer managed) 데이터베이스가 있는 단일 자동화 컨트롤러
- 외부(installer 관리) 데이터베이스가 포함된 단일 자동화 컨트롤러 및 단일 자동화 허브
- 외부(installer managed) 데이터베이스가 포함된 단일 자동화 컨트롤러, 단일 자동화 허브, 단일 이벤트 중심 ansible 컨트롤러 노드

3.1. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 인벤토리 파일 편집

Red Hat Ansible Automation Platform 설치 프로그램 인벤토리 파일을 사용하여 설치 시나리오를 지정할 수 있습니다.

절차

1. 설치 프로그램으로 이동합니다.
   a. [RPM 설치 패키지]

   ```
   $ cd /opt/ansible-automation-platform/installer/
   ```

   b. [bundled installer]

   ```
   $ cd ansible-automation-platform-setup-bundle-<latest-version>
   ```

   c. [online installer]

   ```
   $ cd ansible-automation-platform-setup-<latest-version>
   ```

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 지원되는 설치 시나리오 예제 중 하나를 **인벤토리** 파일의 기반으로 사용할 수 있습니다.

추가 리소스

- Ansible 설치 인벤토리 파일에 사용되는 사전 정의된 변수의 포괄적인 목록은 인벤토리 파일 변수를 참조하십시오.

3.2. 설치 시나리오에 따른 인벤토리 파일 예

Red Hat은 Ansible Automation Platform에 대한 여러 설치 시나리오를 지원합니다. 예제 파일을 기반으로 사용하여 고유한 인벤토리 파일을 개발하거나 선호하는 설치 시나리오에 가장 가까운 예제를 사용할 수 있습니다.

3.2.1. 설치 시나리오에 따른 인벤토리 파일 권장 사항

Ansible Automation Platform에 대한 설치 방법을 선택하기 전에 다음 권장 사항을 검토하십시오. 이러한 권장 사항에 대해 숙지하면 설치 프로세스가 간소화됩니다.

- Red Hat Ansible Automation Platform 또는 자동화 허브: **[automationhub]** 그룹에 자동화 허브 호스트를 추가합니다.
- 프로덕션 또는 고객 환경에서 Ansible Automation Platform 버전에 대해 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치하지 마십시오. 이로 인해 경합 문제 및 리소스 사용량이 많을 수 있습니다.
- **[automationhub]** 및 **[automationcontroller]** 호스트에 연결할 수 있는 IP 주소 또는 FQDN(정규화된 도메인 이름)을 제공하여 사용자가 다른 노드에서 자동화 허브에서 콘텐츠를 동기화하고 설치할 수 있도록 합니다.
  FQDN은 올바르게 처리되지 않으므로 **-** 또는 **_** 기호를 포함하지 않아야 합니다.

  localhost 를 사용하지 마십시오.
- **admin** 은 Ansible Automation Platform에 처음 로그인하는 기본 사용자 ID이며 인벤토리 파일에서 변경할 수 없습니다.
- **pg_password** 에 대한 특수 문자 사용은 제한됩니다. **!**, **#**, **0** 및 **@** 문자가 지원됩니다. 다른 특수 문자를 사용하면 설정이 실패할 수 있습니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service 계정 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.
- 인벤토리 파일 변수 **registry_username** 및 **registry_password** 는 번들 이외의 설치 프로그램을 사용하는 경우에만 필요합니다.

3.2.1.1. 외부(installer 관리) 데이터베이스가 있는 단일 자동화 컨트롤러

이 예제를 사용하여 Red Hat Ansible Automation Platform을 설치할 인벤토리 파일을 채웁니다. 이 설치 인벤토리 파일에는 별도의 노드에 외부 데이터베이스가 있는 단일 자동화 컨트롤러 노드가 포함됩니다.

```
[automationcontroller]
controller.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port=5432
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
```

```
registry_username='<registry username>'
registry_password='<registry password>'

# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.1.2. 외부(installer 관리) 데이터베이스가 포함된 단일 자동화 컨트롤러 및 단일 자동화 허브

이 예제를 사용하여 인벤토리 파일을 작성하여 자동화 컨트롤러 및 자동화 허브의 단일 인스턴스를 외부(installer 관리) 데이터베이스와 함께 배포합니다.

```
[automationcontroller]
controller.example.com

[automationhub]
automationhub.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# The default install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
```

```
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key
```

3.2.1.2.1. Red Hat Single Sign-On 환경에 자동화 허브 연결

자동화 허브를 Red Hat Single Sign-On 설치에 연결하도록 인벤토리 파일을 추가로 구성할 수 있습니다.

Ansible Automation Platform에서 관리하는 Red Hat Single Sign-On 설치에 연결할 때 외부 Red Hat Single Sign-On 설치에 연결할 때와 다른 변수 세트를 구성해야 합니다.

이러한 인벤토리 변수에 대한 자세한 내용은 Ansible Automation Platform의 중앙 인증 설치 및 구성을 참조하십시오.

3.2.1.2.2. 고가용성 자동화 허브

다음 예제를 사용하여 인벤토리 파일을 채우고 고가용성 자동화 허브를 설치합니다. 이 인벤토리 파일에는 클러스터형 설정이 포함된 고가용성 자동화 허브가 포함되어 있습니다.

Red Hat Single Sign-On을 구현하도록 HA 배포를 추가로 구성하고 SELinux에서 자동화 허브의 고가용성 배포를 활성화할 수 있습니다.

데이터베이스 호스트 IP 지정

- **automation_pg_host** 및 **automation_pg_port** 인벤토리 변수를 사용하여 데이터베이스 호스트의 IP 주소를 지정합니다. 예를 들면 다음과 같습니다.

```
automationhub_pg_host='192.0.2.10'
automationhub_pg_port=5432
```

- 또한 **automationhub_pg_host** 인벤토리 변수의 값을 사용하여 [database] 섹션에 데이터베이스 호스트의 IP 주소를 지정합니다.

```
[database]
192.0.2.10
```

클러스터형 설정의 모든 인스턴스 나열

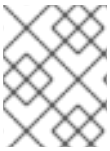
- 클러스터형 설정을 설치하는 경우 [automationhub] 섹션의 **localhost ansible_connection=local** 을 모든 인스턴스의 호스트 이름 또는 IP로 바꿉니다. 예를 들면 다음과 같습니다.

```
[automationhub]
automationhub1.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.18
automationhub2.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.20
automationhub3.testing.ansible.com ansible_user=cloud-user ansible_host=192.0.2.22
```

다음 단계

각 프라이빗 자동화 허브 서버의 **/etc/pulp/settings.py** 지시문이 있는지 확인합니다.

```
USE_X_FORWARDED_PORT = True
USE_X_FORWARDED_HOST = True
```



참고

automationhub_main_url 이 지정되지 않은 경우 [automationhub] 그룹의 첫 번째 노드가 기본값으로 사용됩니다.

3.2.1.2.3. SELinux에서 Automation Hub의 HA(고가용성) 배포 활성화

SELinux에서 자동화 허브의 고가용성 배포를 활성화하도록 인벤토리 파일을 구성할 수 있습니다. /var/lib/pulp 및 **/var/lib/pulp /pulpcore_static** 에 대해 두 개의 마운트 지점을 생성한 다음 각각에 적절한 SELinux 컨텍스트를 할당해야 합니다.

참고

/var/lib/pulp pulpcore_static의 컨텍스트를 추가하고 **/var/lib/pulp**.에 대한 컨텍스트를 추가하기 전에 Ansible Automation Platform 설치 프로그램을 실행해야 합니다.

사전 요구 사항

- 서버에 NFS 내보내기를 이미 구성했습니다.

절차

1. **/var/lib/pulp** 에 마운트 지점을 만듭니다.

```
$ mkdir /var/lib/pulp/
```

2. 텍스트 편집기를 사용하여 **/etc/fstab** 를 열고 다음 값을 추가합니다.

```
srv_rhel8:/data /var/lib/pulp nfs defaults,_netdev,nosharecache 0 0
srv_rhel8:/data/pulpcore_static /var/lib/pulp/pulpcore_static nfs
defaults,_netdev,nosharecache,context="system_u:object_r:httpd_sys_content_rw_t:s0" 0 0
```

3. 다시 로드 systemd 관리자 구성 명령을 실행합니다.

```
$ systemctl daemon-reload
```

4. **/var/lib/pulp**:에 대한 마운트 명령을 실행하십시오.

```
$ mount /var/lib/pulp
```

5. **/var/lib/pulp/pulpcore_static** 에 마운트 지점을 만듭니다.

```
$ mkdir /var/lib/pulp/pulpcore_static
```

6. mount 명령을 실행합니다.

```
$ mount -a
```

7. 마운트 지점이 설정된 경우 Ansible Automation Platform 설치 프로그램을 실행합니다.

```
$ setup.sh -- -b --become-user root
```

8. 설치가 완료되면 **/var/lib/pulp/** 마운트 지점을 마운트 해제합니다.

다음 단계

1. 적절한 SELinux 컨텍스트를 적용합니다.
2. pulpcore.serivce를 구성합니다.

추가 리소스

- SELinux 컨텍스트 목록은 Pulp 프로젝트 설명서의 SELinux 요구 사항을 참조하십시오.
- Pulp 폴더에 대한 자세한 설명은 Filesystem Layout 을 참조하십시오.

3.2.1.2.4. pulpcore.service 구성

인벤토리 파일을 구성하고 SELinux 컨텍스트를 적용한 후 이제 pulp 서비스를 구성해야 합니다.

절차

1. 두 개의 마운트 지점이 설정된 경우 Pulp 서비스를 종료하여 **pulpcore.service** 를 구성합니다.

```
$ systemctl stop pulpcore.service
```

2. **systemctl** 을 사용하여 **pulpcore.service** 를 편집합니다.

```
$ systemctl edit pulpcore.service
```

3. 다음 항목을 **pulpcore.service** 에 추가하여 자동화 허브 서비스가 네트워크를 시작하고 원격 마운트 지점을 마운트한 후에만 시작되도록 합니다.

```
[Unit]
After=network.target var-lib-pulp.mount
```

4. **remote-fs.target** 을 활성화합니다.

```
$ systemctl enable remote-fs.target
```

5. 시스템을 재부팅합니다.

```
$ systemctl reboot
```

문제 해결

pulpcore SELinux 정책의 버그로 **etc/pulp/certs/** 에서 토큰 인증 공개/개인 키가 적절한 SELinux 레이블이 없으므로 pulp 프로세스가 실패할 수 있습니다. 이 경우 다음 명령을 실행하여 적절한 레이블을 임시로 연결합니다.

```
$ chcon system_u:object_r:pulpcore_etc_t:s0 /etc/pulp/certs/token_{private,public}_key.pem
```

이 명령을 반복하여 시스템의 레이블을 다시 지정할 때마다 적절한 SELinux 레이블을 다시 첨부합니다.

3.2.1.2.5. SELinux 컨텍스트 적용

인벤토리 파일을 구성한 후 SELinux에서 자동화 허브의 HA(고가용성) 배포를 활성화하려면 컨텍스트를 적용해야 합니다.

절차

1. Pulp 서비스를 종료합니다.

```
$ systemctl stop pulpcore.service
```

2. **/var/lib/pulp/pulpcore_static** 을 마운트 해제합니다.

```
$ umount /var/lib/pulp/pulpcore_static
```

3. **/var/lib/pulp/**:을 마운트 해제합니다.

```
$ umount /var/lib/pulp/
```

4. 텍스트 편집기를 사용하여 **/etc/fstab** 를 열고 **/var/lib/pulp** 의 기존 값을 다음으로 바꿉니다.

```
srv_rhel8:/data /var/lib/pulp nfs
defaults,_netdev,nosharecache,context="system_u:object_r:pulpcore_var_lib_t:s0" 0 0
```

5. mount 명령을 실행합니다.

```
$ mount -a
```

3.2.1.3. 프라이빗 자동화 허브에서 콘텐츠 서명 구성

Ansible 인증 콘텐츠 컬렉션에 성공적으로 서명하고 게시하려면 서명을 위해 프라이빗 자동화 허브를 구성해야 합니다.

사전 요구 사항

- GnuPG 키 쌍은 조직에서 안전하게 설정 및 관리합니다.
- 공개-개인 키 쌍은 프라이빗 자동화 허브에서 콘텐츠 서명을 구성하기 위한 적절한 액세스 권한을 갖습니다.

절차

1. 파일 이름만 허용하는 서명 스크립트를 생성합니다.

참고

이 스크립트는 서명 서비스 역할을 하며 **PULP_SIGNING_KEY_FINGERPRINT** 환경 변수를 통해 지정된 키를 사용하여 해당 파일의 ascii-armored detached **gpg** 서명을 생성해야 합니다.

스크립트는 다음 형식을 사용하여 JSON 구조를 출력합니다.

```
{"file": "filename", "signature": "filename.asc"}
```

모든 파일 이름은 현재 작업 디렉터리 내부의 상대 경로입니다. 분리된 서명에는 파일 이름이 동일해야 합니다.

예제:

다음 스크립트는 콘텐츠에 대한 서명을 생성합니다.

```
#!/usr/bin/env bash

FILE_PATH=$1
SIGNATURE_PATH="$1.asc"

ADMIN_ID="$PULP_SIGNING_KEY_FINGERPRINT"
PASSWORD="password"

# Create a detached signature
gpg --quiet --batch --pinentry-mode loopback --yes --passphrase \
  $PASSWORD --homedir ~/.gnupg/ --detach-sign --default-key $ADMIN_ID \
  --armor --output $SIGNATURE_PATH $FILE_PATH

# Check the exit status
STATUS=$?
if [ $STATUS -eq 0 ]; then
  echo {\"file\": \"$FILE_PATH\", \"signature\": \"$SIGNATURE_PATH\"}
else
  exit $STATUS
fi
```

Ansible Automation Platform 클러스터에 서명이 활성화된 프라이빗 자동화 허브를 배포하면 컬렉션에 새로운 UI 추가 기능이 표시됩니다.

2. **automationhub_*** 로 시작하는 옵션은 Ansible Automation Platform 설치 프로그램 인벤토리 파일을 검토하십시오.

```
[all:vars]
```

```
.
.
.
automationhub_create_default_collection_signing_service = True
automationhub_auto_sign_collections = True
automationhub_require_content_approval = True
automationhub_collection_signing_service_key = /abs/path/to/galaxy_signing_service.gpg
automationhub_collection_signing_service_script = /abs/path/to/collection_signing.sh
```

두 개의 새 키(**automationhub_auto_sign_collections** 및 Automation **hub_require_content_approval**)는 컬렉션이 프라이빗 자동화 허브에 업로드된 후 서명하고 승인되어야 함을 나타냅니다.

3.2.1.4. 프라이빗 자동화 허브에서 LDAP 구성

LDAP 인증에 대한 프라이빗 자동화 허브를 구성하려면 Red Hat Ansible Automation Platform 설치 프로그램 인벤토리 파일에서 다음 6개의 변수를 설정해야 합니다.

- **automationhub_authentication_backend**
- **automationhub_ldap_server_uri**
- **automationhub_ldap_bind_dn**
- **automationhub_ldap_bind_password**
- **automationhub_ldap_user_search_base_dn**
- **automationhub_ldap_group_search_base_dn**

이러한 변수 중 하나라도 누락된 경우 Ansible Automation 설치 프로그램에서 설치를 완료할 수 없습니다.

3.2.1.4.1. 인벤토리 파일 변수 설정

LDAP 인증을 사용하여 프라이빗 자동화 허브를 구성할 때 설치 프로세스 중에 인벤토리 파일에 적절한 변수를 설정해야 합니다.

절차

1. Red Hat Ansible Automation Platform 설치 프로그램 인벤토리 파일 편집 절차에 따라 인벤토리 파일에 액세스합니다.
2. 다음 예제를 가이드로 사용하여 Ansible Automation Platform 인벤토리 파일을 설정합니다.

```
automationhub_authentication_backend = "ldap"

automationhub_ldap_server_uri = "ldap://ldap:389"   (for LDAPs use
automationhub_ldap_server_uri = "ldaps://ldap-server-fqdn")
automationhub_ldap_bind_dn = "cn=admin,dc=ansible,dc=com"
automationhub_ldap_bind_password = "GoodNewsEveryone"
automationhub_ldap_user_search_base_dn = "ou=people,dc=ansible,dc=com"
automationhub_ldap_group_search_base_dn = "ou=people,dc=ansible,dc=com"
```

참고

다른 옵션으로 설정하지 않는 한 다음 변수는 기본값으로 설정됩니다.

```
auth_ldap_user_search_scope= 'SUBTREE'
auth_ldap_user_search_filter= '(uid=%(user)s)'
auth_ldap_group_search_scope= 'SUBTREE'
auth_ldap_group_search_filter= '(objectClass=Group)'
auth_ldap_group_type_class= 'django_auth_ldap.config:GroupOfNamesType'
```

3. 선택 사항: 개인 자동화 허브에서 사용자 그룹, 슈퍼유저 액세스 또는 미러링과 같은 추가 매개변수를 설정합니다. Configure extra LDAP 매개변수로 이동하여 이 선택적 단계를 완료합니다.

3.2.1.4.2. 추가 LDAP 매개변수 구성

슈퍼유저 액세스, 사용자 그룹, 미러링 또는 기타 추가 매개변수를 설정하려는 경우 **ldap_extra_settings** 사전에 구성하는 YAML 파일을 생성할 수 있습니다.

절차

1. **ldap_extra_settings** 가 포함된 YAML 파일을 생성합니다.
   - 예제:

```
#ldapextras.yml
---
ldap_extra_settings:
 <LDAP_parameter>: <Values>

...
```

2. 설정에 필요한 매개변수를 추가합니다. 다음 예제에서는 **ldap_extra_settings** 에서 설정할 수 있는 LDAP 매개변수를 설명합니다.
   - 이 예제를 사용하여 LDAP 그룹의 멤버십에 따라 슈퍼유저 플래그를 설정합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_FLAGS_BY_GROUP: {"is_superuser": "cn=pah-
admins,ou=groups,dc=example,dc=com",}
...
```

   - 이 예제를 사용하여 슈퍼유저 액세스를 설정합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_FLAGS_BY_GROUP: {"is_superuser": "cn=pah-
admins,ou=groups,dc=example,dc=com",}
...
```

   - 이 예제를 사용하여 사용자가 속한 모든 LDAP 그룹을 미러링합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_MIRROR_GROUPS: True
...
```

- 이 예제를 사용하여 LDAP 사용자 속성(예: 사용자의 이름, 성, 이메일 주소)을 매핑합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_USER_ATTR_MAP: {"first_name": "givenName", "last_name": "sn",
"email": "mail",}
...
```

- 다음 예제를 사용하여 LDAP 그룹 멤버십에 따라 액세스 권한을 부여하거나 거부합니다.
  - 프라이빗 자동화 허브 액세스 권한을 부여하려면(예: **cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** 그룹)

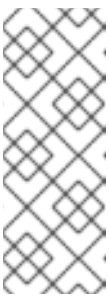
```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_REQUIRE_GROUP: 'cn=pah-
nosoupforyou,ou=groups,dc=example,dc=com'
...
```

  - 프라이빗 자동화 허브 액세스를 거부하려면 (예: **cn=pah-nosoupforyou,ou=groups,dc=example,dc=com** 그룹)의 멤버:

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_DENY_GROUP: 'cn=pah-
nosoupforyou,ou=groups,dc=example,dc=com'
...
```

- LDAP 디버그 로깅을 활성화하려면 이 예제를 사용합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  GALAXY_LDAP_LOGGING: True
...
```



참고

setup.sh 를 다시 실행하는 것이 바람직하지 않거나 디버그 로깅이 짧은 시간 동안 활성화된 경우 프라이빗 자동화 허브의 **/etc/pulp/settings.py** 파일에 **GALAXY_LDAP_LOGGING: True** 를 포함하는 행을 수동으로 추가할 수 있습니다. 변경 사항을 적용하려면 **pulpcore-api.service** 및 **nginx.service** 를 모두 다시 시작합니다. 인적 오류로 인한 실패를 방지하려면 필요한 경우에만 이 방법을 사용하십시오.

- **AUTH_LDAP_CACHE_TIMEOUT** 변수를 설정하여 LDAP 캐싱을 구성하려면 이 예제를 사용합니다.

```
#ldapextras.yml
---
ldap_extra_settings:
  AUTH_LDAP_CACHE_TIMEOUT: 3600
...
```

3. 개인 자동화 허브 설치 중에 **setup.sh -e @ldapextras.yml** 을 실행합니다. .Verification 올바르게 설정되어 있는지 확인하려면 프라이빗 자동화 허브의 **/etc/pulp/settings.py** 파일의 모든 설정을 볼 수 있는지 확인하십시오.

3.2.1.4.3. LDAP 참조

LDAP 서버에서 추천을 반환하는 경우 프라이빗 자동화 허브에서 LDAP를 사용하여 성공적으로 인증하기 위해 추천을 비활성화해야 할 수 있습니다.

그렇지 않으면 다음 메시지가 반환됩니다.

```
Operation unavailable without authentication
```

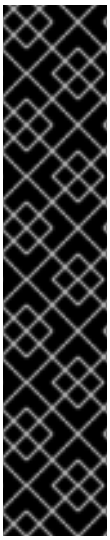
LDAP REFERRALS 조회를 비활성화하려면 다음을 설정합니다.

```
GALAXY_LDAP_DISABLE_REFERRALS = true
```

이렇게 하면 **AUTH_LDAP_CONNECTIONS_OPTIONS** 가 올바른 옵션으로 설정됩니다.

3.2.1.5. 외부(installer 관리) 데이터베이스가 포함된 단일 자동화 컨트롤러, 단일 자동화 허브, 단일 이벤트 기반 Ansible 컨트롤러 노드

이 예제를 사용하여 인벤토리 파일을 채우고 자동화 컨트롤러, 자동화 허브 및 이벤트 기반 Ansible 컨트롤러의 단일 인스턴스를 외부(installer managed) 데이터베이스로 배포합니다.



중요

- 이 시나리오에는 이벤트 기반 Ansible 컨트롤러를 성공적으로 배포하려면 최소 자동화 컨트롤러 2.4가 필요합니다.
- 이벤트 기반 Ansible 컨트롤러는 별도의 서버에 설치해야 하며 자동화 허브 및 자동화 컨트롤러와 동일한 호스트에 설치할 수 없습니다.
- 표준 조건에서 이벤트 기반 Ansible 룰북을 활성화하면 약 250MB의 메모리를 사용합니다. 그러나 실제 메모리 사용은 규칙의 복잡성과 처리된 이벤트의 볼륨 및 크기에 따라 크게 다를 수 있습니다. 많은 수의 이벤트가 예상되거나 룰북 복잡성이 높은 시나리오에서는 스테이징 환경에서 리소스 사용을 사전 평가하십시오. 이렇게 하면 최대 활성화 수가 리소스 용량을 기반으로 합니다. 다음 예에서 기본 **automationedacontroller_max_running_activations** 설정은 12이지만 용량에 따라 조정할 수 있습니다.

```
[automationcontroller]
controller.example.com
```

```
[automationhub]
automationhub.example.com

[automationedacontroller]
automationedacontroller.example.com

[database]
data.example.com

[all:vars]
admin_password='<password>'
pg_host='data.example.com'
pg_port='5432'
pg_database='awx'
pg_username='awx'
pg_password='<password>'
pg_sslmode='prefer'  # set to 'verify-full' for client-side enforced SSL

registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

# {HubNameStart} configuration

automationhub_admin_password= <PASSWORD>

automationhub_pg_host='data.example.com'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password=<PASSWORD>
automationhub_pg_sslmode='prefer'

# Automation {EDAController} configuration

automationedacontroller_admin_password='<eda-password>'

automationedacontroller_pg_host='data.example.com'
automationedacontroller_pg_port=5432

automationedacontroller_pg_database='automationedacontroller'
automationedacontroller_pg_username='automationedacontroller'
automationedacontroller_pg_password='<password>'

# Keystore file to install in SSO node
# sso_custom_keystore_file='/path/to/sso.jks'

# This install will deploy SSO with sso_use_https=True
# Keystore password is required for https enabled SSO
sso_keystore_password=''

# This install will deploy a TLS enabled Automation Hub.
# If for some reason this is not the behavior wanted one can
# disable TLS enabled deployment.
#
```

```
# automationhub_disable_https = False
# The default install will generate self-signed certificates for the Automation
# Hub service. If you are providing valid certificate via automationhub_ssl_cert
# and automationhub_ssl_key, one should toggle that value to True.
#
# automationhub_ssl_validate_certs = False
# SSL-related variables
# If set, this will install a custom CA certificate to the system trust store.
# custom_ca_cert=/path/to/ca.crt
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.key

# Certificate and key to install in nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Server-side SSL settings for PostgreSQL (when we are installing it).
# postgres_use_ssl=False
# postgres_ssl_cert=/path/to/pgsql.crt
# postgres_ssl_key=/path/to/pgsql.key

# Boolean flag used to verify Automation Controller's
# web certificates when making calls from Automation {EDAcontroller}.
# automationedacontroller_controller_verify_ssl = true
#
# Certificate and key to install in Automation {EDAcontroller} node
# automationedacontroller_ssl_cert=/path/to/automationeda.crt
# automationedacontroller_ssl_key=/path/to/automationeda.key
```

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

개인 자동화 허브 설치에 필요한 매개 변수로 인벤토리 파일을 업데이트한 후 설치 프로그램 설정 스크립트를 실행합니다.

절차

- **setup.sh** 스크립트 실행

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

Red Hat Ansible Automation Platform 설치가 시작됩니다.

3.4. 자동화 컨트롤러 설치 확인

인벤토리 파일에 삽입한 관리자 인증 정보로 로그인하여 자동화 컨트롤러를 성공적으로 설치했는지 확인합니다.

사전 요구 사항

- 포트 443 사용 가능

절차

1. **인벤토리** 파일의 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. 사용자 ID **admin** 및 **인벤토리** 파일에 설정한 암호 자격 증명을 사용하여 로그인합니다.

참고

자동화 컨트롤러 서버는 포트 80(**https://<CONTROLLER_SERVER_NAME>/**)에서 액세스할 수 있지만 포트 443으로 리디렉션됩니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.4 설치가 완료됩니다.

3.4.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 3.1. 자동화 컨트롤러를 구성하는 리소스

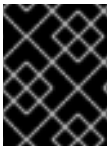
리소스 링크	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러를 설정하고 첫 번째 플레이북을 실행합니다.
자동화 컨트롤러 관리 가이드	고객 스크립트, 관리 작업 등을 통해 자동화 컨트롤러 관리를 구성합니다.
Red Hat Ansible Automation Platform에 대한 프록시 지원 구성	프록시 서버를 사용하여 자동화 컨트롤러를 설정합니다.
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보를 관리합니다.
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능을 보다 자세히 검토합니다.

3.5. 자동화 허브 설치 확인

인벤토리 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 허브를 성공적으로 설치했는지 확인합니다.

절차

1. **인벤토리** 파일에서 자동화 허브 노드에 지정된 IP 주소로 이동합니다.
2. 사용자 ID **admin** 및 **인벤토리** 파일에 설정한 암호 자격 증명을 사용하여 로그인합니다.



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 을 통해 Ansible에 문의하십시오.

자동화 허브에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.4 설치가 완료됩니다.

3.5.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 3.2. 자동화 컨트롤러를 구성하는 리소스

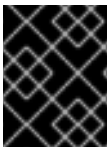
리소스 링크	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스를 구성합니다.
자동화 허브에서 Red Hat Certified, validated, Ansible Galaxy 콘텐츠 관리	자동화 허브에 콘텐츠를 추가합니다.
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션을 게시합니다.

3.6. 이벤트 기반 ANSIBLE 컨트롤러 설치 확인

인벤토리 파일에 삽입한 관리자 인증 정보로 로그인하여 Event-Driven Ansible 컨트롤러를 성공적으로 설치했는지 확인합니다.

절차

1. **인벤토리** 파일의 이벤트 기반 Ansible 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. 사용자 ID **admin** 및 **인벤토리** 파일에 설정한 암호 자격 증명을 사용하여 로그인합니다.



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 을 통해 Ansible에 문의하십시오.

이벤트 기반 Ansible 컨트롤러에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.4 설치가 완료됩니다.

4장. 연결이 해제된 설치

인터넷에 연결되지 않았거나 온라인 리포지토리에 액세스할 수 없는 경우 활성 인터넷 연결 없이 Red Hat Ansible Automation Platform을 설치할 수 있습니다.

4.1. 사전 요구 사항

연결이 끊긴 네트워크에 Ansible Automation Platform을 설치하기 전에 다음 사전 요구 사항을 충족해야 합니다.

1. 생성된 서브스크립션 매니페스트입니다. 자세한 내용은 매니페스트 파일 가져오기를 참조하십시오.
2. 고객 포털 의 Ansible Automation Platform 설치 번들이 다운로드됩니다.
3. 자동화 컨트롤러 및 프라이빗 자동화 허브 서버의 DNS 레코드 가 생성됩니다.

4.2. 연결이 끊긴 RHEL에 ANSIBLE AUTOMATION PLATFORM 설치

자동화 컨트롤러에 있는 설치 관리자 관리 데이터베이스를 사용하여 인터넷 연결 없이 Ansible Automation Platform 자동화 컨트롤러 및 프라이빗 자동화 허브를 설치할 수 있습니다. 연결이 끊긴 환경에서 Ansible Automation Platform을 더 쉽게 설치할 수 있는 추가 구성 요소가 포함되어 있으므로 연결이 끊긴 설치에 대한 설치 번들을 사용합니다. 여기에는 Ansible Automation Platform RPM(Red Hat 패키지 관리자) 및 기본 실행 환경(EE) 이미지가 포함됩니다.

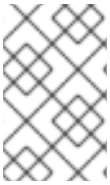
4.2.1. 연결이 끊긴 설치를 위한 시스템 요구 사항

Ansible Automation Platform의 연결이 끊긴 설치를 수행하기 전에 시스템에 모든 하드웨어 요구 사항이 있는지 확인합니다. 하드웨어 요구 사항에 대한 자세한 내용은 Chapter 2를 참조하십시오. 시스템 요구 사항.

4.2.2. RPM Source

BaseOS 및 AppStream 리포지토리에서 제공되는 Ansible Automation Platform용 RPM 종속 항목은 설정 번들에 포함되어 있지 않습니다. 이러한 종속 항목을 추가하려면 먼저 BaseOS 및 AppStream 리포지토리에 대한 액세스 권한을 가져와야 합니다. Satellite를 사용하여 리포지토리를 동기화하고 종속성을 추가합니다. 대체 툴을 선호하는 경우 다음 옵션 중에서 선택할 수 있습니다.

- Reposync
- RHEL 바이너리 DVD



참고

RHEL Binary DVD 방법에는 버전 8.6 이상을 포함하여 지원되는 RHEL 버전에 대한 DVD가 필요합니다. 현재 지원되는 RHEL 버전에 대한 정보는 Red Hat Enterprise Linux 라이프사이클 을 참조하십시오.

추가 리소스

- Satellite

4.3. REPOSYNC를 사용하여 RPM 리포지토리 동기화

reposync를 수행하려면 인터넷에 액세스할 수 있는 RHEL 호스트가 필요합니다. 리포지토리를 동기화한 후 리포지토리를 웹 서버에서 호스팅되는 연결이 끊긴 네트워크로 이동할 수 있습니다.

절차

1. BaseOS 및 AppStream 필수 리포지토리를 연결합니다.

```
# subscription-manager repos \
    --enable rhel-8-for-x86_64-baseos-rpms \
    --enable rhel-8-for-x86_64-appstream-rpms
```

2. reposync를 수행합니다.

```
# dnf install yum-utils
# reposync -m --download-metadata --gpgcheck \
    -p /path/to/download
```

   a. **--download-metadata** 와 **--newest-only** 없이 reposync를 사용합니다. RHEL 8 Reposync를 참조하십시오.

      - **--newest-only를 사용하지 않는 경우** 다운로드한 리포지토리는 ~90GB가 됩니다.
      - **--newest-only를 사용하는 경우** 다운로드한 리포지토리는 ~14GB가 됩니다.

3. Red Hat Single Sign-On을 사용하려는 경우 다음 리포지토리를 동기화합니다.

   a. jb-eap-7.3-for-rhel-8-x86_64-rpms

   b. rh-sso-7.4-for-rhel-8-x86_64-rpms

   reposync가 완료되면 리포지토리를 웹 서버와 함께 사용할 준비가 된 것입니다.

4. 리포지토리를 연결이 끊긴 네트워크로 이동합니다.

4.4. 리포지토리를 호스팅할 새 웹 서버 생성

리포지토리를 호스팅할 기존 웹 서버가 없는 경우 동기화된 리포지토리를 사용하여 리포지토리를 생성할 수 있습니다.

절차

1. 사전 요구 사항 설치:

```
$ sudo dnf install httpd
```

2. 리포지토리 디렉터리를 제공하도록 httpd를 구성합니다.

```
/etc/httpd/conf.d/repository.conf

DocumentRoot '/path/to/repos'

<LocationMatch "^/+$">
    Options -Indexes
    ErrorDocument 403 /.noindex.html
</LocationMatch>
```

```
<Directory '/path/to/repos'>
    Options All Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

3. apache 사용자가 디렉터리를 읽을 수 있는지 확인합니다.

```
$ sudo chown -R apache /path/to/repos
```

4. SELinux를 구성합니다.

```
$ sudo semanage fcontext -a -t httpd_sys_content_t "/path/to/repos(/.*)?"
$ sudo restorecon -ir /path/to/repos
```

5. httpd를 활성화합니다.

```
$ sudo systemctl enable --now httpd.service
```

6. 방화벽을 엽니다.

```
$ sudo firewall-cmd --zone=public --add-service=http –add-service=https --permanent
$ sudo firewall-cmd --reload
```

7. 자동화 컨트롤러 및 자동화 허브에서 리포지토리 파일을 */etc/yum.repos.d/local.repo* 에 추가하고 필요한 경우 선택적 리포지토리를 추가합니다.

```
[Local-BaseOS]
name=Local BaseOS
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-baseos-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[Local-AppStream]
name=Local AppStream
baseurl=http://<webserver_fqdn>/rhel-8-for-x86_64-appstream-rpms
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

4.5. 로컬에 마운트된 DVD에서 RPM 리포지토리에 액세스

RHEL 바이너리 DVD에서 리포지토리에 액세스하려면 먼저 로컬 리포지토리를 설정해야 합니다.

절차

1. DVD 또는 ISO 마운트:

   a. DVD

```
# mkdir /media/rheldvd && mount /dev/sr0 /media/rheldvd
```

b. ISO

```
# mkdir /media/rheldvd && mount -o loop rhrhel-8.6-x86_64-dvd.iso /media/rheldvd
```

2. **/etc/yum.repos.d/ECDHE.repo에 yum repo 파일을 만듭니다.**

```
[dvd-BaseOS]
name=DVD for RHEL - BaseOS
baseurl=file:///media/rheldvd/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

[dvd-AppStream]
name=DVD for RHEL - AppStream
baseurl=file:///media/rheldvd/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

3. gpg 키를 가져옵니다.

```
# rpm --import /media/rheldvd/RPM-GPG-KEY-redhat-release
```

참고

키를 가져오지 않으면 다음과 유사한 오류가 표시됩니다.

```
# Curl error (6): Couldn't resolve host name for
https://www.redhat.com/security/data/fd431d51.txt [Could not resolve host:
www.redhat.com]
```

추가 리소스

리포지토리 설정에 대한 자세한 내용은 Red Hat Enterprise Linux 8에서 로컬로 마운트된 DVD에 대한 yum 리포지토리를 설정해야 합니다.

4.6. 인터넷 연결 없이 ANSIBLE AUTOMATION PLATFORM에 서브스크립션 매니페스트 추가

인터넷 연결 없이 Ansible Automation Platform에 서브스크립션을 추가하려면 서브스크립션 매니페스트를 생성하고 가져옵니다.

절차

1. Red Hat 고객 포털에 로그인합니다.
2. 메뉴 표시줄에서 서브스크립션을 **선택하고 서브스크립션 할당 탭을** 선택합니다.
3. **새 서브스크립션 할당** 을 클릭합니다.
4. 새 서브스크립션 할당의 이름을 지정합니다.

5. **유형** 목록에서 **Satellite 6.8** 을 선택합니다.
6. **생성**을 클릭합니다. 서브스크립션 할당에 대한 **세부 정보** 탭이 열립니다.
7. **서브스크립션 탭을** 선택합니다.
8. **서브스크립션 추가를** 클릭합니다.
9. Ansible Automation Platform 서브스크립션을 찾고 **인타이틀먼트** 상자에서 환경에 할당할 인타이틀먼트 수를 **추가합니다**. Ansible Automation Platform(서버, 네트워크 장치 등)에서 관리할 각 노드에 대해 단일 인타이틀먼트가 필요합니다.
10. **Submit** 을 클릭합니다.
11. **Export Manifest** 를 클릭합니다.

이렇게 하면 설치 후 자동화 컨트롤러로 가져오는 파일 *manifest_<allocation name>_<date>.zip* 이 다운로드됩니다.

4.7. ANSIBLE AUTOMATION PLATFORM 설치 번들 다운로드 및 설치

연결 해제된 설치를 위해 Ansible Automation Platform을 다운로드할 설치 번들을 선택합니다. 이 번들에는 Ansible Automation Platform용 RPM 콘텐츠 및 설치 프로세스 중에 프라이빗 자동화 허브에 업로드할 기본 실행 환경 이미지가 포함되어 있습니다.

절차

1. Red Hat Ansible Automation Platform 다운로드 페이지로 이동한 후 Ansible Automation Platform 2.4 설치 번들에 대한 **지금 다운로드를** 클릭하여 Ansible Automation Platform 설치 번들 패키지를 다운로드합니다.
2. 자동화 컨트롤러에서 번들의 압축을 풉니다.

```
$ tar xvf \
  ansible-automation-platform-setup-bundle-2.4-1.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.4-1
```

3. 필요한 옵션을 포함하도록 인벤토리 파일을 편집합니다.
   a. Automationcontroller 그룹
   b. automationhub 그룹
   c. admin_password
   d. pg_password
   e. automationhub_admin_password
   f. automationhub_pg_host, automationhub_pg_port
   g. automationhub_pg_password

   인벤토리 파일의 예

```
[automationcontroller]
automationcontroller.example.org ansible_connection=local
```

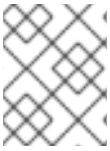
```
[automationcontroller:vars]
peers=execution_nodes

[automationhub]
automationhub.example.org

[all:vars]
admin_password='password123'

pg_database='awx'
pg_username='awx'
pg_password='dbpassword123'

receptor_listener_port=27199

automationhub_admin_password='hubpassword123'

automationhub_pg_host='automationcontroller.example.org'
automationhub_pg_port=5432

automationhub_pg_database='automationhub'
automationhub_pg_username='automationhub'
automationhub_pg_password='dbpassword123'
automationhub_pg_sslmode='prefer'
```

4. Ansible Automation Platform 설치 번들 실행 파일을 root 사용자로 실행합니다.

```
$ sudo -i
# cd /path/to/ansible-automation-platform-setup-bundle-2.4-1
# ./setup.sh
```

5. 설치가 완료되면 설치 인벤토리 파일에 지정된 자동화 컨트롤러 노드의 FQDN(정규화된 도메인 이름)으로 이동합니다.

6. 설치 인벤토리 파일에 지정된 관리자 자격 증명을 사용하여 로그인합니다.



참고

백업, 복원 및 업그레이드 기능에 사용되므로 인벤토리 파일을 설치 후 그대로 유지해야 합니다. 인벤토리 파일에 암호가 포함된 경우 백업 사본을 안전한 위치에 보관합니다.

4.8. 사후 설치 작업 완료

Ansible Automation Platform 설치를 완료한 후 자동화 허브 및 자동화 컨트롤러가 올바르게 배포되었는지 확인합니다.

4.8.1. 컨트롤러 서브스크립션 추가

절차

1. 자동화 컨트롤러의 FQDN으로 이동합니다. 인벤토리 파일에서 사용자 이름 admin 및 **admin_password** 로 지정한 암호로 로그인합니다.

2. **찾아보기를** 클릭하고 이전에 만든 *manifest.zip* 을 선택합니다.
3. **다음**을 클릭합니다.
4. **사용자 분석** 및 **자동화 분석** 확인 . 이들은 인터넷 연결에 의존하며 꺼야 합니다.
5. **다음**을 클릭합니다.
6. 최종 사용자 라이센스 계약을 읽고 동의한 경우 **제출을** 클릭합니다.

4.8.2. CA 신뢰 저장소 업데이트

설치 후 작업의 일부로 소프트웨어의 인증서를 업데이트해야 합니다. 기본적으로 Ansible Automation Platform 자동화 허브 및 자동화 컨트롤러는 자체 서명된 인증서를 사용하여 설치됩니다. 이로 인해 컨트롤러는 허브의 인증서를 신뢰하지 않으며 허브에서 실행 환경을 다운로드하지 않습니다.

자동화 컨트롤러에서 자동화 허브에서 실행 환경을 다운로드하려면 허브의 CA(인증 기관) 인증서를 컨트롤러의 신뢰할 수 있는 인증서로 가져와야 합니다. 자동화 컨트롤러와 프라이빗 자동화 허브 간에 SSH를 root 사용자로 사용할 수 있는지 여부에 따라 두 가지 방법 중 하나로 이 작업을 수행할 수 있습니다.

4.8.2.1. root 사용자로 보안 복사(SCP) 사용

SSH를 컨트롤러와 프라이빗 자동화 허브 간에 root 사용자로 사용할 수 있는 경우 SCP를 사용하여 프라이빗 자동화 허브의 루트 인증서를 컨트롤러에 복사합니다.

절차

1. 컨트롤러에서 **update-ca-trust** 를 실행하여 CA 신뢰 저장소를 업데이트합니다.

```
$ sudo -i
# scp <hub_fqdn>:/etc/pulp/certs/root.crt
/etc/pki/ca-trust/source/anchors/automationhub-root.crt
# update-ca-trust
```

4.8.2.2. root가 아닌 사용자로 복사 및 붙여넣기

개인 자동화 허브와 컨트롤러 간에 SSH를 root로 사용할 수 없는 경우 개인 자동화 허브의 */etc/pulp/certs/root.crt* 파일의 내용을 복사하여 */etc/pki/ca-trust/source/anchors/automationhub-root.crt* 라는 컨트롤러의 새 파일에 붙여넣습니다.

절차

1. **update-ca-trust** 를 실행하여 새 인증서로 CA 신뢰 저장소를 업데이트합니다. 프라이빗 자동화 허브에서 다음을 실행합니다.

```
$ sudo -i
# cat /etc/pulp/certs/root.crt
(copy the contents of the file, including the lines with 'BEGIN CERTIFICATE' and
'END CERTIFICATE')
```

1. 자동화 컨트롤러에서 다음을 수행합니다.

```
$ sudo -i
# vi /etc/pki/ca-trust/source/anchors/automationhub-root.crt
```

```
(paste the contents of the root.crt file from the private automation hub into the new file and write to
disk)
# update-ca-trust
```

추가 리소스

- 알 수 없는 인증 기관에 대한 자세한 내용은 Ansible Automation Platform 2.1의 알 수 없는 인증 기관 오류와 함께 프로젝트 동기화 실패를 참조하십시오.

4.9. 프라이빗 자동화 허브로 컬렉션 가져오기

개인 자동화 허브에서 사용할 수 있도록 Ansible 자동화 허브에서 tarball 파일로 컬렉션을 다운로드할 수 있습니다. 인증 컬렉션은 자동화 허브 하이브리드 클라우드 콘솔에서 사용할 수 있으며 커뮤니티 컬렉션은 Ansible Galaxy 에서 사용할 수 있습니다. 또한 컬렉션에 필요한 모든 종속성을 다운로드하여 설치해야 합니다.

절차

1. link: console.redhat.com 으로 이동하여 Red Hat 인증 정보를 사용하여 로그인합니다.
2. 다운로드할 **컬렉션을** 클릭합니다.
3. **tarball 다운로드를**클릭합니다.
4. 컬렉션에 종속성이 있는지 확인하려면 종속성 탭을 **클릭합니다**.
5. 이 컬렉션에 필요한 모든 종속성을 다운로드합니다.

4.10. 컬렉션 네임스페이스 생성

컬렉션을 가져오기 전에 먼저 개인 자동화 허브에서 컬렉션의 네임스페이스를 생성해야 합니다. tarball 파일 이름의 첫 번째 부분을 보면 네임스페이스 이름을 찾을 수 있습니다. 예를 들어 컬렉션 *ansible-netcommon-3.0.0.tar.gz* 의 네임스페이스는 *ansible* 입니다.

절차

1. 자동화 허브 하이브리드 클라우드 콘솔에 로그인합니다.
2. 탐색 패널에서 **컬렉션 → 네임스페이스를** 선택합니다.
3. **생성을** 클릭합니다.
4. 네임스페이스 이름을 제공합니다.
5. **생성을** 클릭합니다.

4.10.1. 웹 콘솔을 사용하여 tarball 컬렉션 가져오기

네임스페이스가 생성되면 웹 콘솔을 사용하여 컬렉션을 가져올 수 있습니다.

절차

1. 자동화 허브 하이브리드 클라우드 콘솔에 로그인합니다.

2. 탐색 패널에서 **컬렉션 → 네임스페이스를** 선택합니다.
3. **컬렉션을 가져올 네임스페이스 옆에 있는 컬렉션 보기를** 클릭합니다.
4. **Upload collection** 을 클릭합니다.
5. **폴더 아이콘을** 클릭하고 컬렉션의 tarball을 선택합니다.
6. **업로드**를 클릭합니다.

그러면 '내 가져오기' 페이지가 열립니다. 가져온 파일 및 모듈의 가져오기 상태 및 다양한 세부 정보를 확인할 수 있습니다.

4.10.2. CLI를 사용하여 tarball 컬렉션 가져오기

GUI 대신 명령줄 인터페이스를 사용하여 컬렉션을 프라이빗 자동화 허브로 가져올 수 있습니다.

절차

1. 컬렉션 tarballs를 프라이빗 자동화 허브에 복사합니다.
2. SSH를 통해 프라이빗 자동화 허브 서버에 로그인합니다.
3. 자동화 허브의 신뢰 저장소에 자체 서명된 루트 CA 인증서를 추가합니다.

```
# cp /etc/pulp/certs/root.crt \
    /etc/pki/ca-trust/source/anchors/automationhub-root.crt
# update-ca-trust
```

4. 자동화 허브 구성으로 **/etc/ansible/ansible.cfg** 파일을 업데이트합니다. 인증에 토큰 또는 사용자 이름 및 암호를 사용합니다.

```
[galaxy]
server_list = private_hub

[galaxy_server.private_hub]
url=https://<hub_fqdn>/api/galaxy/
token=<token_from_private_hub>
```

5. ansible-gal Galaxyy 명령을 사용하여 컬렉션을 가져옵니다.

```
$ ansible-galaxy collection publish <collection_tarball>
```

4.11. 가져온 컬렉션 승인

GUI 또는 CLI 방법을 사용하여 컬렉션을 가져온 후에는 GUI를 사용하여 컬렉션을 승인해야 합니다. 승인되면 사용할 수 있습니다.

절차

1. 자동화 허브 하이브리드 클라우드 콘솔에 로그인합니다.
2. 탐색 패널에서 **컬렉션 → 승인** 을 선택합니다.

3. **승인하려는 컬렉션에 대해 승인을** 클릭합니다.
4. 이제 컬렉션을 프라이빗 자동화 허브에서 사용할 수 있습니다.
5. 2단계와 3단계를 반복하여 컬렉션에 대한 종속성을 가져옵니다.

참고

컬렉션은 소스에 관계없이 "Published" 리포지토리에 추가됩니다.

권장 컬렉션은 사용 사례에 따라 다릅니다. Ansible과 Red Hat은 이러한 컬렉션을 제공합니다.

4.11.1. 사용자 정의 자동화 실행 환경

ansible-builder 프로그램을 사용하여 사용자 지정 실행 환경 이미지를 생성합니다. 연결이 끊긴 환경의 경우 다음과 같은 방법으로 사용자 정의 실행 환경 이미지를 빌드할 수 있습니다.

- 인터넷 연결 시스템에서 실행 환경 이미지를 빌드하고 연결이 끊긴 환경으로 가져옵니다.
- ansible-builder를 사용하는 일반 프로세스를 일부 수정하여 연결이 끊긴 환경에서 완전히 실행 환경 이미지를 빌드합니다.
- 연결이 끊긴 환경에 필요한 모든 수정이 포함된 최소 기본 컨테이너 이미지를 생성한 다음 기본 컨테이너 이미지에서 사용자 정의 실행 환경 이미지를 빌드합니다.

4.11.1.1. 연결이 끊긴 경계에서 사용자 정의 실행 환경 이미지 전송

인터넷에 연결된 시스템에서 사용자 정의 실행 환경 이미지를 빌드할 수 있습니다. 실행 환경을 생성한 후 로컬 podman 이미지 캐시에서 사용할 수 있습니다. 그런 다음 연결이 끊긴 경계에서 사용자 정의 실행 환경 이미지를 전송할 수 있습니다.

절차

1. 이미지를 저장합니다.

```
$ podman image save localhost/custom-ee:latest | gzip -c custom-ee-latest.tar.gz
```

   sneakernet 또는 일방 다이어와 같은 기존 메커니즘을 사용하여 연결이 끊긴 경계에서 파일을 전송합니다.

2. 연결이 끊긴 쪽에서 이미지를 사용할 수 있는 후 로컬 podman 캐시로 가져와서 태그를 지정하여 연결 해제된 허브로 푸시합니다.

```
$ podman image load -i custom-ee-latest.tar.gz
$ podman image tag localhost/custom-ee <hub_fqdn>/custom-ee:latest
$ podman login <hub_fqdn> --tls-verify=false
$ podman push <hub_fqdn>/custom-ee:latest
```

4.12. 연결이 끊긴 환경에서 실행 환경 빌드

Ansible Automation Platform의 실행 환경 생성 은 연결이 끊긴 환경에서 다르게 작동하는 일반적인 작업입니다. 사용자 정의 실행 환경을 빌드할 때 ansible-builder 툴은 기본적으로 인터넷의 다음 위치에서 콘텐츠를 다운로드합니다.

- 실행 환경 이미지에 추가된 모든 Ansible 콘텐츠 컬렉션에 대한 Red Hat Automation Hub(console.redhat.com) 또는 Ansible Galaxy(galaxy.ansible.com)입니다.
- 컬렉션 종속성으로 필요한 python 패키지의 경우 PyPI(pypi.org)입니다.
- 필요한 경우 RPM을 실행 환경 이미지에 추가하거나 업데이트하기 위한 RHEL 또는 UBI 리포지토리(cdn.redhat.com)와 같은 RPM 리포지토리입니다.
- registry.redhat.io는 기본 컨테이너 이미지에 액세스할 수 있습니다.

연결이 끊긴 환경에서 실행 환경 이미지를 빌드하려면 이러한 위치의 콘텐츠를 미러링해야 합니다. Ansible Ansible Galaxy 또는 자동화 허브에서 프라이빗 자동화 허브로 컬렉션을 가져오는 방법에 대한 정보는 프라이빗 자동화 허브로 컬렉션 가져오기를 참조하십시오.

연결이 끊긴 네트워크로 전송된 미러링된 PyPI 콘텐츠는 웹 서버 또는 Nexus와 같은 아티팩트 저장소를 사용하여 사용할 수 있습니다. RHEL 및 UBI 리포지토리 콘텐츠는 인터넷에 연결된 Red Hat Satellite 서버에서 내보낸 다음 연결이 끊긴 Satellite로 복사한 다음 사용자 정의 실행 환경을 빌드하는 데 사용할 수 있도록 할 수 있습니다. 자세한 내용은 Air-Gapped Scenario의 ISS Export Sync 를 참조하십시오.

기본 기본 컨테이너 이미지 ee-minimal-rhel8은 사용자 지정 실행 환경 이미지를 생성하는 데 사용되며 번들 설치 프로그램에 포함됩니다. 이 이미지는 설치 시 프라이빗 자동화 허브에 추가됩니다. ee-minimal-rhel9와 같은 다른 기본 컨테이너 이미지가 필요한 경우 연결이 끊긴 네트워크로 가져와 프라이빗 자동화 허브 컨테이너 레지스트리에 추가해야 합니다.

연결이 끊긴 네트워크에서 모든 사전 요구 사항을 사용할 수 있게 되면 ansible-builder 명령을 사용하여 사용자 정의 실행 환경 이미지를 생성할 수 있습니다.

4.12.1. Ansible Builder RPM 설치

사용자 지정 실행 환경이 빌드되는 RHEL 시스템에서 환경에 이미 존재하는 Satellite 서버를 사용하여 Ansible Builder RPM을 설치합니다. 실행 환경 이미지에서 필요한 경우 기존 Satellite의 모든 RHEL 콘텐츠를 사용할 수 있으므로 이 방법을 사용하는 것이 좋습니다.

절차

1. Ansible Automation Platform 리포지토리에서 Ansible Builder RPM을 설치합니다.
   a. 연결이 끊긴 네트워크의 Satellite에 RHEL 시스템을 서브스크립션합니다.
   b. Ansible Automation Platform 서브스크립션을 연결하고 AAP 리포지토리를 활성화합니다. 리포지토리 이름은 기본 시스템에서 사용되는 RHEL 버전에 따라 **ansible-automation-platform-2.4-for-rhel-8-x86_64-rpms** 또는 **ansible-automation-platform-2.4-for-rhel-9-x86_64-rpms** 입니다.
   c. Ansible Builder RPM을 설치합니다. 아래 예제가 제대로 작동하려면 Ansible Builder RPM 버전이 3.0.0 이상이어야 합니다.
2. Ansible Automation Platform 설치 번들에서 Ansible Builder RPM을 설치합니다. 연결이 끊긴 네트워크에서 Satellite 서버를 사용할 수 없는 경우 이 방법을 사용합니다.
   a. Ansible Automation Platform 설치 번들의 보관을 해제합니다.
   b. 포함된 콘텐츠에서 Ansible Builder RPM 및 해당 종속 항목을 설치합니다.

```
$ tar -xzvf ansible-automation-platform-setup-bundle-2.4-3-x86_64.tar.gz
$ cd ansible-automation-platform-setup-bundle-2.4-3-x86_64/bundle/packages/el8/repos/
```

```
$ sudo dnf install ansible-builder-3.0.0-2.el8ap.noarch.rpm \
    python39-requirements-parser-0.2.0-4.el8ap.noarch.rpm \
    python39-bindep-2.10.2-3.el8ap.noarch.rpm \
    python39-jsonschema-4.16.0-1.el8ap.noarch.rpm \
    python39-pbr-5.8.1-2.el8ap.noarch.rpm \
    python39-distro-1.6.0-3.el8pc.noarch.rpm \
    python39-packaging-21.3-2.el8ap.noarch.rpm \
    python39-parsley-1.3-2.el8pc.noarch.rpm \
    python39-attrs-21.4.0-2.el8pc.noarch.rpm \
    python39-pyrsistent-0.18.1-2.el8ap.x86_64.rpm \
    python39-pyparsing-3.0.9-1.el8ap.noarch.rpm
```

참고

사용 중인 설치 번들 버전에 따라 특정 버전이 약간 다를 수 있습니다.

추가 리소스

- 연결이 끊긴 네트워크에서 Satellite 환경을 만드는 방법에 대한 자세한 내용은 Disconnected Network Environment에서 Satellite Server 설치를 참조하십시오.

4.12.2. 사용자 정의 실행 환경 정의 생성

Ansible Builder RPM이 설치되면 다음 단계를 사용하여 사용자 지정 실행 환경을 생성합니다.

1. 사용자 정의 실행 환경을 생성할 때 사용되는 빌드 아티팩트용 디렉터리를 생성합니다. 아래 단계로 생성된 새 파일은 이 디렉터리에 생성됩니다.

```
$ mkdir $HOME/custom-ee $HOME/custom-ee/files
$ cd $HOME/custom-ee/
```

2. 사용자 정의 실행 환경에 대한 요구 사항을 정의하는 **execution-environment.yml** 파일을 생성합니다.

참고

실행 환경 정의 형식의 버전 3이 필요하므로 계속하기 전에 **execution-environment.yml** 파일에 **버전 3** 이 명시적으로 포함되어 있는지 확인합니다.

   a. 프라이빗 자동화 허브에서 사용할 수 있는 최소 실행 환경을 가리키도록 기본 이미지를 재정의합니다.

   b. 빌드 프로세스에서 사용할 연결이 끊긴 콘텐츠 소스를 가리키는 데 필요한 추가 빌드 파일을 정의합니다. 사용자 정의 **execution-environment.yml** 파일은 다음 예와 유사해야 합니다.

```
$ cat execution-environment.yml
---
version: 3

images:
  base_image:
    name: private-hub.example.com/ee-minimal-rhel8:latest

dependencies:
```

```
  python: requirements.txt
  galaxy: requirements.yml

additional_build_files:
  - src: files/ansible.cfg
    dest: configs
  - src: files/pip.conf
    dest: configs
  - src: files/hub-ca.crt
    dest: configs
  # uncomment if custom RPM repositories are required
  #- src: files/custom.repo
  #  dest: configs

additional_build_steps:
  prepend_base:
    # copy a custom pip.conf to override the location of the PyPI content
    - ADD _build/configs/pip.conf /etc/pip.conf
    # remove the default UBI repository definition
    - RUN rm -f /etc/yum.repos.d/ubi.repo
    # copy the hub CA certificate and update the trust store
    - ADD _build/configs/hub-ca.crt /etc/pki/ca-trust/source/anchors
    - RUN update-ca-trust
    # if needed, uncomment to add a custom RPM repository configuration
    #- ADD _build/configs/custom.repo /etc/yum.repos.d/custom.repo

  prepend_galaxy:
    - ADD _build/configs/ansible.cfg ~/.ansible.cfg

...
```

3. 개인 자동화 허브를 가리키는 **files/** 하위 디렉터리에 **ansible.cfg** 파일을 생성합니다.

```
$ cat files/ansible.cfg
[galaxy]
server_list = private_hub

[galaxy_server.private_hub]
url = https://private-hub.example.com/api/galaxy/
```

4. 내부 PyPI 미러(웹 서버 또는 Nexus와 같은 항목)를 가리키는 **files/** 하위 디렉터리에 **pip.conf** 파일을 생성합니다.

```
$ cat files/pip.conf
[global]
index-url = https://<pypi_mirror_fqdn>/
trusted-host = <pypi_mirror_fqdn>
```

5. 선택 사항: **bindep.txt** 파일을 사용하여 사용자 지정 실행 환경을 추가하는 경우, 연결이 끊긴 Satellite 또는 RPM 리포지토리를 호스팅하는 다른 위치를 가리키는 **files/** 하위 디렉터리에 **custom.repo** 파일을 만듭니다. 이 단계가 필요한 경우 **custom.repo** 파일에 해당하는 **execution-environment.yml** 파일의 단계 주석을 제거합니다.
   다음 예제는 UBI 리포지토리를 위한 것입니다. 다른 로컬 리포지토리도 이 파일에 추가할 수 있습니다. 웹 서버에 미러 콘텐츠가 있는 위치에 따라 URL 경로를 변경해야 할 수 있습니다.

```
$ cat files/custom.repo
[ubi-8-baseos]
name = Red Hat Universal Base Image 8 (RPMs) - BaseOS
baseurl = http://<ubi_mirror_fqdn>/repos/ubi-8-baseos
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1

[ubi-8-appstream]
name = Red Hat Universal Base Image 8 (RPMs) - AppStream
baseurl = http://<ubi_mirror_fqdn>/repos/ubi-8-appstream
enabled = 1
gpgkey = file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
gpgcheck = 1
```

6. 프라이빗 자동화 허브 웹 서버 인증서에 서명하는 데 사용되는 CA 인증서를 추가합니다. 프라이빗 자동화 허브에서 설치 프로그램에서 제공하는 자체 서명된 인증서를 사용하는 경우:

   a. 개인 자동화 허브에서 **/etc/pulp/certs/pulp_webserver.crt** 파일을 복사하여 이름을 **hub-ca.crt** 로 지정합니다.

   b. **hub-ca.crt** 파일을 **files/** 하위 디렉터리에 추가합니다.

7. 프라이빗 자동화 허브에서 인증 기관에서 서명한 사용자 제공 인증서를 사용하는 경우:

   a. 해당 CA 인증서의 사본을 만들고 이름을 **hub-ca.crt** 로 지정합니다.

   b. **hub-ca.crt** 파일을 **files/** 하위 디렉터리에 추가합니다.

8. 이전 단계가 완료되면 사용자 정의 실행 환경 이미지에 필요한 콘텐츠를 사용하여 python **requirements.txt** 및 Ansible collection **requirements.yml** 파일을 생성합니다.

참고

필요한 컬렉션은 이미 프라이빗 자동화 허브에 업로드해야 합니다.

다음 파일은 **bindep.txt** 및 **files/custom.repo** 가 선택인 **custom-ee/** 디렉터리에 있어야 합니다.

```
$ cd $HOME/custom-ee
$ tree .
.
├── bindep.txt
├── execution-environment.yml
├── files
│   ├── ansible.cfg
│   ├── custom.repo
│   ├── hub-ca.crt
│   └── pip.conf
├── requirements.txt
└── requirements.yml

1 directory, 8 files
```

추가 리소스

버전 3 형식 및 요구 사항에 대한 자세한 내용은 실행 환경 정의: 버전 3 형식을 참조하십시오.

4.12.3. 사용자 정의 실행 환경 빌드

새 사용자 정의 실행 환경을 생성하기 전에 콘텐츠를 다운로드하려면 개인 허브의 API 토큰이 필요합니다.

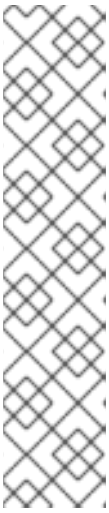
다음 단계를 수행하여 토큰을 생성합니다.

1. 프라이빗 허브에 로그인합니다.
2. 왼쪽 메뉴에서 "Collections"를 선택합니다.
3. 메뉴의 "Collections" 섹션에서"API 토큰"을 선택합니다.
4. 토큰이 있으면 Ansible Builder가 토큰에 액세스할 수 있도록 다음 환경 변수를 설정합니다.

```
$ export ANSIBLE_GALAXY_SERVER_PRIVATE_HUB_TOKEN=<your_token>
```

5. 명령을 사용하여 사용자 정의 실행 환경을 생성합니다.

```
$ cd $HOME/custom-ee
$ ansible-builder build -f execution-environment.yml -t private-hub.example.com/custom-ee:latest -v 3
```



참고

개인 허브 인증서가 알 수 없는 기관에서 서명한 오류로 인해 빌드가 실패하면 명령을 실행하여 필요한 이미지를 로컬 이미지 캐시로 가져올 수 있습니다.

```
$ podman pull private-hub.example.com/ee-minimal-rhel8:latest --tls-verify=false
```

또는 podman 인증서 저장소에 개인 허브 CA 인증서를 추가할 수 있습니다.

```
$ sudo mkdir /etc/containers/certs.d/private-hub.example.com
$ sudo cp $HOME/custom-ee/files/hub-ca.crt /etc/containers/certs.d/private-hub.example.com
```

4.12.4. 프라이빗 자동화 허브에 사용자 정의 실행 환경 업로드

새 실행 환경 이미지를 자동화 작업에 사용하려면 먼저 프라이빗 자동화 허브에 업로드해야 합니다.

먼저 실행 환경 이미지가 로컬 podman 캐시에 표시되는지 확인합니다.

```
$ podman images --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID      REPOSITORY                    TAG
b38e3299a65e private-hub.example.com/custom-ee        latest
8e38be53b486 private-hub.example.com/ee-minimal-rhel8  latest
```

그런 다음 프라이빗 자동화 허브의 컨테이너 레지스트리에 로그인하고 이미지를 푸시하여 작업 템플릿 및 워크플로우와 함께 사용할 수 있도록 합니다.

```
$ podman login private-hub.example.com -u admin
```

```
Password:
Login Succeeded!
$ podman push private-hub.example.com/custom-ee:latest
```

4.13. 마이너 ANSIBLE AUTOMATION PLATFORM 릴리스 간 업그레이드

Ansible Automation Platform 2의 마이너 릴리스 간에 업그레이드하려면 이 일반 워크플로우를 사용하십시오.

절차

1. 최신 Ansible Automation Platform 2 설치 번들을 다운로드하여 보관 해제합니다.
2. 기존 설치의 백업을 생성합니다.
3. 기존 설치 인벤토리 파일을 새 설치 번들 디렉터리에 복사합니다.
4. **./setup.sh** 를 실행하여 설치를 업그레이드합니다.

예를 들어 버전 2.2.0-7에서 2.3-1.2로 업그레이드하려면 설치가 발생한 초기 컨트롤러 노드에 두 설치 번들이 있는지 확인합니다.

```
 $ ls -1F
ansible-automation-platform-setup-bundle-2.2.0-7/
ansible-automation-platform-setup-bundle-2.2.0-7.tar.gz
ansible-automation-platform-setup-bundle-2.3-1.2/
ansible-automation-platform-setup-bundle-2.3-1.2.tar.gz
```

2.2.0-7 설치를 백업하십시오.

```
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ sudo ./setup.sh -b
$ cd ..
```

2.2.0-7 인벤토리 파일을 2.3-1.2 번들 디렉터리에 복사합니다.

```
$ cd ansible-automation-platform-setup-bundle-2.2.0-7
$ cp inventory ../ansible-automation-platform-setup-bundle-2.3-1.2/
$ cd ..
```

setup.sh 스크립트를 사용하여 2.2.0-7에서 2.3-1.2로 업그레이드하십시오.

```
$ cd ansible-automation-platform-setup-bundle-2.3-1.2
$ sudo ./setup.sh
```

부록 A. 인벤토리 파일 변수

다음 표에는 Ansible 설치 인벤토리 파일에 사용되는 사전 정의된 변수에 대한 정보가 포함되어 있습니다. 이러한 변수가 모두 필요한 것은 아닙니다.

A.1. 일반 변수

변수	description
enable_insights_collection	기본 설치는 노드가 Subscription Manager에 등록된 경우 Red Hat Insights for Red Hat Ansible Automation Platform Service에 노드를 등록합니다. 비활성화하려면 **False** 로 설정합니다. default = **true**
nginx_user_http_config	http 섹션의 **/etc/nginx/nginx.conf** 에 대한 nginx 구성 목록입니다. 목록의 각 요소는 별도의 행으로 **http nginx config** 에 제공됩니다. default = 빈 목록
registry_password	**registry_password** 는 번들 이외의 설치 프로그램을 사용하는 경우에만 필요합니다. **registry_url** 에 액세스하기 위한 암호 인증 정보 . **[automationcontroller]** 및 **[automationhub]** 그룹 모두에 사용됩니다. **registry_username** 및 **registry_password** 에 Red Hat Registry Service 계정 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다. **registry_url** 이 **registry.redhat.io** 인 경우 번들 설치 프로그램을 사용하지 않는 경우 사용자 이름과 암호가 필요합니다.
registry_url	**[automationcontroller]** 및 **[automationhub]** 그룹 모두에 사용됩니다. default = **registry.redhat.io**.

변수	description
registry_username	**registry_username** 은 번들 이외의 설치 프로그램을 사용하는 경우에만 필요합니다. **registry_url** 에 액세스하기 위한 사용자 인증 정보 . **[automationcontroller]** 및 **[automationhub]** 그룹 모두에 사용되지만 **registry_url** 값이 **registry.redhat.io** 인 경우에만 사용됩니다. **registry_username** 및 **registry_password** 에 Red Hat Registry Service 계정 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.
routable_hostname	**라우팅 가능한 호스트** 이름은 설치 프로그램을 실행하는 시스템이 특정 URL을 통해서만 라우팅할 수 있는 경우 사용됩니다. 예를 들어 인벤토리에서 단축 이름을 사용하는 경우, 설치 프로그램을 실행하는 노드는 FQDN을 사용하여 해당 호스트를 확인할 수 있습니다. **routable_hostname** 이 설정되지 않은 경우 기본적으로 **ansible_host** 로 설정되어야 합니다. **ansible_host** 를 설정하지 않으면**inventory_hostname** 이 마지막 수단으로 사용됩니다. 이 변수는 **[all:vars]** 섹션이 아닌 특정 호스트의 호스트 변수로 사용됩니다. 자세한 내용은 하나의 machine:host 변수에 변수 할당을 참조하십시오.

A.2. ANSIBLE 자동화 허브 변수

변수	description
automationhub_admin_password	필수 항목 인벤토리 파일에 일반 텍스트로 제공되면 암호를 따옴표로 묶어야 합니다.
automationhub_api_token	Ansible Automation Platform 2.0 이하에서 업그레이드하는 경우 다음 중 하나를 수행해야 합니다. ● 기존 Ansible 자동화 허브 토큰을 **automationhub_api_token** 로 제공하거나 ● **generate_automationhub_token** 을 **true** 로 설정하여 새 토큰을 생성합니다. 새 토큰을 생성하면 기존 토큰이 무효화됩니다.

변수	description
automationhub_authentication_backend	이 변수는 기본적으로 설정되어 있지 않습니다. LDAP 인증을 사용하려면 **ldap** 로 설정합니다. 이 값이 **ldap** 로 설정된 경우 다음 변수도 설정해야 합니다. ● **automationhub_ldap_server_uri** ● **automationhub_ldap_bind_dn** ● **automationhub_ldap_bind_password** ● **automationhub_ldap_user_search_base_dn** ● **automationhub_ldap_group_search_base_dn** 이러한 항목이 없으면 설치가 중지됩니다.
automationhub_auto_sign_collections	컬렉션 서명 서비스가 활성화된 경우 컬렉션은 기본적으로 자동으로 서명되지 않습니다. 이 매개변수를 **true** 로 설정하면 기본적으로 서명합니다. default = **false** 입니다.
automationhub_backup_collections	*선택 사항* Ansible 자동화 허브는 **/var/lib/pulp** 에 아티팩트를 제공합니다. 자동화 컨트롤러는 기본적으로 아티팩트를 자동으로 백업합니다. **automationhub_backup_collections** 를 false로 설정하면 백업/복원 프로세스가 **/var/lib/pulp** 를 백업하거나 복원하지 않습니다. default = **true**.
automationhub_collection_download_count	*선택 사항* 다운로드 수가 UI에 표시되는지 여부를 확인합니다. default = **false** 입니다.

변수	description
automationhub_collection_seed_repository	번들 설치 프로그램을 실행하면 검증된 콘텐츠가 **검증된** 리포지토리에 업로드되고 인증된 콘텐츠가**rh-certified** repository에 업로드됩니다. 기본적으로 인증된 콘텐츠와 검증된 콘텐츠가 모두 업로드됩니다. 이 변수의 가능한 값은 '인증' 또는 'validated'입니다. 콘텐츠를 설치하지 않으려면 **automationhub_seed_collections** 를 **false** 로 설정하여 시드를 비활성화합니다. 하나의 유형의 콘텐츠만 원한다면 **automationhub_seed_collections** 를 **true** 로 설정하고 **automationhub_collection_seed_repository** 를 포함하려는 콘텐츠 유형으로 설정합니다.
automationhub_collection_signing_service_key	컬렉션 서명 서비스가 활성화된 경우 컬렉션을 올바르게 서명할 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/key/to/sign**
automationhub_collection_signing_service_script	컬렉션 서명 서비스가 활성화된 경우 컬렉션을 올바르게 서명할 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	컬렉션 서명 서비스를 생성하려면 이 변수를 true로 설정합니다. default = **false** 입니다.
automationhub_container_signing_service_key	컨테이너 서명 서비스가 활성화된 경우 컨테이너를 올바르게 서명할 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/key/to/sign**
automationhub_container_signing_service_script	컨테이너 서명 서비스가 활성화된 경우 컨테이너를 올바르게 서명할 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/script/that/signs**
automationhub_create_default_container_signing_service	컨테이너 서명 서비스를 생성하려면 이 변수를 true로 설정합니다. default = **false** 입니다.

변수	description
automationhub_disable_hsts	기본 설치는 TLS가 활성화된 Ansible 자동화 허브를 배포합니다. HSTS(*HTTP Strict Transport Security*) 웹 보안 정책이 활성화된 자동화 허브를 배포하는 경우 이 변수를 사용합니다. 이 변수는 HSTS 웹 보안 정책 메커니즘을 비활성화합니다. default = **false** 입니다.
automationhub_disable_https	*선택 사항* Ansible 자동화 허브가 HTTPS가 활성화된 경우 배포됨. default = **false** 입니다.
automationhub_enable_api_access_log	**true** 로 설정하면 이 변수는 사용자 이름 및 IP 주소를 포함하여 모든 사용자 작업을 플랫폼에 기록하는 **/var/log/galaxy_api_access.log** 에 로그 파일을 생성합니다. default = **false** 입니다.
automationhub_enable_analytics	Ansible Automation Platform 2.4의 자동화 허브에 사용되는 pulpcore 버전에 대한 pulp analytics를 활성화할지 여부를 나타내는 부울입니다. pulp 분석을 활성화하려면 **automationhub_enable_analytics** 를 true로 설정합니다. default = **false** 입니다.
automationhub_enable_unauthenticated_collection_access	권한이 없는 사용자가 컬렉션을 볼 수 있도록 하려면 이 변수를 true로 설정합니다. default = **false** 입니다.
automationhub_enable_unauthenticated_collection_download	권한이 없는 사용자가 컬렉션을 다운로드할 수 있도록 하려면 이 변수를 true로 설정합니다. default = **false** 입니다.

변수	description
automationhub_importer_settings	*선택 사항* galvncy-importer에 전달할 설정 사전입니다. 가져오기 시 컬렉션은 일련의 검사를 수행할 수 있습니다. 동작은 galvncy **-importer.cfg 구성으로** 구동됩니다. 예를 들면 **ansible-doc**,**ansible-lint**, **flake8** 이 있습니다. 이 매개변수를 사용하면 이 구성을 구동할 수 있습니다.
automationhub_main_url	클라이언트가 연결하는 주요 자동화 허브 URL입니다. 예를 들면 https://<load balancer host>입니다. **automationhub_main_url** 을 사용하여 자동화 허브 환경에서 Red Hat Single Sign-On을 구현하는 경우 클라이언트가 연결하는 기본 자동화 허브 URL을 지정합니다. 지정하지 않으면 **[automationhub]** 그룹의 첫 번째 노드가 사용됩니다.
automationhub_pg_database	*필수 항목* 데이터베이스 이름입니다. 기본값 = Automation **hub**.
automationhub_pg_host	내부 데이터베이스를 사용하지 않는 경우 필요합니다. 자동화 허브에서 사용하는 원격 PostgreSQL 데이터베이스의 호스트 이름입니다. default = **127.0.0.1**.
automationhub_pg_password	자동화 허브 PostgreSQL 데이터베이스의 암호입니다. Automation **hub_pg_password** 에는 특수 문자 사용이 제한됩니다. **!**, **#**, **0** 및 **@** 문자가 지원됩니다. 다른 특수 문자를 사용하면 설정이 실패할 수 있습니다.
automationhub_pg_port	내부 데이터베이스를 사용하지 않는 경우 필요합니다. 기본값 = 5432.
automationhub_pg_sslmode	필수 항목입니다. default = **prefer**.

변수	description
automationhub_pg_username	필수 항목 기본값 = Automation **hub**.
automationhub_require_content_approval	*선택 사항* 자동화 허브에서 컬렉션을 사용할 수 있기 전에 승인 메커니즘을 적용하는 경우 값이 **true** 입니다. 기본적으로 컬렉션을 자동화 허브에 업로드할 때 관리자는 사용자가 사용할 수 있게 되기 전에 승인해야 합니다. 콘텐츠 승인 흐름을 비활성화하려면 변수를 **false** 로 설정합니다. default = **true**.
automationhub_seed_collections	사전 로드 여부를 정의하는 부울입니다. 번들 설치 프로그램을 실행하면 검증된 콘텐츠가 **검증된** 리포지토리에 업로드되고 인증된 콘텐츠가**rh-certified** repository에 업로드됩니다. 기본적으로 인증된 콘텐츠와 검증된 콘텐츠가 모두 업로드됩니다. 콘텐츠를 설치하지 않으려면 **automationhub_seed_collections** 를 **false** 로 설정하여 시드를 비활성화합니다. 하나의 유형의 콘텐츠만 원한다면 **automationhub_seed_collections** 를 **true** 로 설정하고 **automationhub_collection_seed_repository** 를 포함하려는 콘텐츠 유형으로 설정합니다. default = **true**.
automationhub_ssl_cert	*선택 사항* **/path/to/automationhub.cert** 는 **web_server_ssl_cert** 와 동일하지만 자동화 허브 UI 및 API의 경우입니다.
automationhub_ssl_key	*선택 사항* **/path/to/automationhub.key**. **web_server_ssl_key** 와 동일하지만 자동화 허브 UI 및 API의 경우

변수	description
automationhub_ssl_validate_certs	Red Hat Ansible Automation Platform 2.2 이상에서는 이 값이 더 이상 사용되지 않습니다. Ansible Automation Platform은 기본적으로 자체 서명 인증서로 배포되므로 자동화 허브가 자체적으로 인증서의 유효성을 검사해야 하는 경우 value를 **true** 로 설정합니다. default = **false** 입니다.
automationhub_upgrade	**더 이상 사용되지 않음** Ansible Automation Platform 2.2.1 이상의 경우 이 값은 **true** 로 수정되었습니다. 자동화 허브는 항상 최신 패키지로 업데이트됩니다.
automationhub_user_headers	Ansible 자동화 허브의 웹 서버에 대한 nginx 헤더 목록입니다. 목록의 각 요소는 웹 서버의 nginx 구성에 별도의 행으로 제공됩니다. default = 빈 목록
ee_from_hub_only	자동화 허브를 사용하여 배포하면 설치 프로그램이 실행 환경 이미지를 자동화 허브로 푸시하고 자동화 허브 레지스트리에서 이미지를 가져오도록 자동화 컨트롤러를 구성합니다. 자동화 허브를 실행 환경 이미지를 가져오는 유일한 레지스트리로 설정하려면 이 변수를 **true** 로 설정합니다. **false** 로 설정하면 실행 환경 이미지도 Red Hat에서 직접 가져옵니다. 번들 설치 프로그램을 사용하는 경우 default = **true** 입니다.
generate_automationhub_token	Red Hat Ansible Automation Platform 2.0 또는 이전 버전에서 업그레이드하는 경우 다음 옵션 중 하나를 선택합니다. ● 기존 Ansible 자동화 허브 토큰을 **automationhub_api_token**으로 제공 ● **generate_automationhub_token** 을 **true** 로 설정하여 새 토큰을 생성합니다. 새 토큰을 생성하면 기존 토큰이 무효화됩니다.

변수	description
nginx_hsts_max_age	이 변수는 시스템을 HSM(*HTTP Strict Transport Security*) 호스트로 간주해야 하는 시간(초)을 지정합니다. 즉, HTTPS가 통신에만 사용되는 시간입니다. 기본값은 63072000초 또는 2년입니다.
nginx_tls_protocols	Nginx의 **ssl_protocols** 지원을 정의합니다. 사용 가능한 **TLSv1**,**TLSv1.1, 'TLSv1.2**,**TLSv1.3** TLSv1.1 및 TLSv1.2 매개변수는 OpenSSL 1.0.1 이상이 사용되는 경우에만 작동합니다. TLSv1.3 매개변수는 OpenSSL 1.1.1 이상이 사용되는 경우에만 작동합니다. **nginx_tls-protocols = ['TLSv1.3']** 만 TLSv1.3이 활성화된 경우. 둘 이상의 프로토콜을 설정하려면 **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']**을 사용합니다. Default = **TLSv1.2**.
pulp_db_fields_key	가져올 대칭 암호화 키의 상대 경로 또는 절대 경로입니다. 경로는 Ansible 관리 노드에 있습니다. 자격 증명과 같은 데이터베이스의 특정 필드를 암호화하는 데 사용됩니다. 지정하지 않으면 새 키가 생성됩니다.
sso_automation_platform_login_theme	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. 요소 파일이 있는 디렉터리의 경로입니다. 이 변수를 변경하는 경우 고유한 topic 파일을 제공해야 합니다. 기본값 = **ansible-automation-platform**.
sso_automation_platform_realm	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. SSO의 영역 이름입니다. 기본값 = **ansible-automation-platform**.

변수	description
sso_automation_platform_realm_displayname	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. 영역의 표시 이름입니다. 기본값 = **Ansible Automation Platform**.
sso_console_admin_username	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. SSO 관리 사용자 이름. 기본값 = **admin**.
sso_console_admin_password	*필수 항목* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. SSO 관리 암호.
sso_custom_keystore_file	*선택 사항* Ansible Automation Platform에서 관리되는 Red Hat Single Sign-On에만 사용됩니다. SSO용 고객 제공 키 저장소입니다.
sso_host	*필수 항목* 외부에서 관리되는 Red Hat Single Sign-On에만 사용됩니다. 자동화 허브에는 인증을 위해 SSO 및 SSO 관리 자격 증명이 필요합니다. SSO가 구성에 대해 인벤토리에 제공되지 않으면 이 변수를 사용하여 SSO 호스트를 정의해야 합니다.
sso_keystore_file_remote	*선택 사항* Ansible Automation Platform에서 관리되는 Red Hat Single Sign-On에만 사용됩니다. 고객 제공 키 저장소가 원격 노드에 있는 경우 **true** 로 설정합니다. default = **false** 입니다.

변수	description
sso_keystore_name	*선택 사항* Ansible Automation Platform에서 관리되는 Red Hat Single Sign-On에만 사용됩니다. SSO의 키 저장소 이름입니다. 기본값 = **ansible-automation-platform**.
sso_keystore_password	HTTPS가 활성화된 SSO의 키 저장소 암호입니다. Ansible Automation Platform 관리 SSO 및 HTTPS가 활성화된 경우 필수 항목입니다. 기본 설치는 **sso_use_https=true** 를 사용하여 SSO를 배포합니다.
sso_redirect_host	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. **sso_redirect_host** 가 설정된 경우 애플리케이션에서 인증을 위해 SSO에 연결하는 데 사용됩니다. 클라이언트 시스템에서 액세스할 수 있어야 합니다.
sso_ssl_validate_certs	*선택 사항* Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. 연결 중에 인증서를 검증해야 하는 경우 **true** 로 설정합니다. default = **true**.
sso_use_https	*선택 사항* Single Sign On에서 HTTPS를 사용하는 경우 Ansible Automation Platform에서 관리 및 외부에서 관리되는 Red Hat Single Sign-On에 사용됩니다. default = **true**.

LDAP에 직접 연결하려면 다음 변수를 구성해야 합니다. **ldap_extra_settings** 변수를 사용하여 전달할 수 있는 추가 LDAP 관련 변수 목록은 Django 참조 설명서를 참조하십시오.

변수	description

변수	description
automationhub_ldap_bind_dn	**automationhub_ldap_bind_password** 를 사용하여 LDAP 서버에 바인딩할 때 사용할 이름입니다. 프라이빗 자동화 허브를 LDAP와 통합할 때 설정해야 합니다. 그렇지 않으면 설치에 실패합니다.
automationhub_ldap_bind_password	*필수 항목* **automationhub_ldap_bind_dn** 과 함께 사용할 암호입니다. 프라이빗 자동화 허브 LDAP를 통합할 때 설정해야 합니다. 그렇지 않으면 설치에 실패합니다.
automationhub_ldap_group_search_base_dn	사용자가 속할 수 있는 모든 LDAP 그룹을 찾는 LDAP Search 오브젝트입니다. 구성에서 LDAP 그룹에 대한 참조를 생성하는 경우 이 변수와 **automationhub_ldap_group_type** 을 설정해야 합니다. 프라이빗 자동화 허브를 LDAP와 통합할 때 설정해야 합니다. 그렇지 않으면 설치에 실패합니다. 기본값 = **없음**
automationhub_ldap_group_search_filter	*선택 사항* 그룹 멤버십을 찾으려면 필터를 검색합니다. variable은 자동화 허브 및 LDAP를 사용하여 매핑 그룹에 사용할 objectClass 유형을 식별합니다. LDAP를 사용하여 자동화 허브를 설치하는 데 사용됩니다. Default = **(objectClass=Group)**
automationhub_ldap_group_search_scope	*선택 사항* LDAP 인증에 django 프레임워크를 사용하여 LDAP 트리에서 그룹을 검색할 범위입니다. LDAP를 사용하여 자동화 허브를 설치하는 데 사용됩니다. default = **SUBTREE**

변수	description
automationhub_ldap_group_type	**automationhub_ldap_group_search** 에서 반환된 그룹 유형을 설명합니다. 이는 **automationhub_ldap_group_type_params** 및 **automationhub_group_type_class** 값을 기반으로 동적으로 설정됩니다. 그렇지 않으면 'None'인 django-ldap에서 제공되는 기본값입니다. Default = **django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_group_type_class	*선택 사항* django-ldap 그룹 유형 클래스의 가져오기 가능한 경로입니다. variable은 LDAP 인증을 위한 django 프레임 워크 내에서 그룹 검색 중에 사용되는 그룹 유형을 식별합니다. LDAP를 사용하여 자동화 허브를 설치하는 데 사용됩니다. Default =**django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_server_uri	LDAP 서버의 URI입니다. 기본 LDAP 라이브러리에서 지원하는 모든 URI를 사용합니다. 프라이빗 자동화 허브 LDAP를 통합할 때 설정해야 합니다. 그렇지 않으면 설치에 실패합니다.
automationhub_ldap_user_search_base_dn	디렉터리에서 사용자를 찾는 LDAP Search 오브젝트입니다. filter 매개변수에는 사용자 이름의 자리 표시자 %(user)가 포함되어야 합니다. 인증이 성공하려면 정확히 하나의 결과를 반환해야 합니다. 프라이빗 자동화 허브를 LDAP와 통합할 때 설정해야 합니다. 그렇지 않으면 설치에 실패합니다.
automationhub_ldap_user_search_filter	*선택 사항* Default = **'(uid=%(user)s)'**

변수	description
automationhub_ldap_user_search_scope	*선택 사항* LDAP 인증에 대해 Django 프레임워크를 사용하여 LDAP 트리에서 사용자를 검색하는 범위입니다. LDAP를 사용하여 자동화 허브를 설치하는 데 사용됩니다. default = **SUBTREE**

A.3. 자동화 컨트롤러 변수

변수	description
admin_password	설치가 완료되면 관리 사용자가 UI에 액세스할 수 있는 암호입니다. 인벤토리 파일에 일반 텍스트로 제공되면 암호를 따옴표로 묶어야 합니다.
automation_controller_main_url	SSO 구성에 필요한 대체 프런트 엔드 URL의 경우 URL을 제공합니다.
automationcontroller_password	자동화 컨트롤러 인스턴스의 암호입니다. 인벤토리 파일에 일반 텍스트로 제공되면 암호를 따옴표로 묶어야 합니다.
automationcontroller_username	자동화 컨트롤러 인스턴스의 사용자 이름입니다.
nginx_http_port	nginx HTTP 서버는 인바운드 연결을 수신 대기합니다. 기본값 = 80
nginx_https_port	nginx HTTPS 서버는 보안 연결을 수신 대기합니다. 기본값 = 443
nginx_hsts_max_age	이 변수는 시스템을 HSTS(*HTTP Strict Transport Security*) 호스트로 간주해야 하는 기간을 초 단위로 지정합니다. 즉, HTTPS가 통신에만 사용되는 시간입니다. 기본값은 63072000초 또는 2년입니다.

변수	description
nginx_tls_protocols	Nginx의 **ssl_protocols** 지원을 정의합니다. 사용 가능한 **TLSv1**,**TLSv1.1, 'TLSv1.2**,**TLSv1.3** TLSv1.1 및 TLSv1.2 매개변수는 OpenSSL 1.0.1 이상이 사용되는 경우에만 작동합니다. TLSv1.3 매개변수는 OpenSSL 1.1.1 이상이 사용되는 경우에만 작동합니다. **nginx_tls-protocols = ['TLSv1.3']** 만 TLSv1.3이 활성화된 경우. 둘 이상의 프로토콜을 설정하려면 **nginx_tls_protocols = ['TLSv1.2', 'TLSv.1.3']**을 사용합니다. Default = **TLSv1.2**.
nginx_user_headers	자동화 컨트롤러 웹 서버의 nginx 헤더 목록입니다. 목록의 각 요소는 웹 서버의 nginx 구성에 별도의 행으로 제공됩니다. default = 빈 목록
node_state	*선택 사항* 노드 또는 노드 그룹의 상태입니다. 유효한 옵션은 **활성 상태** 이거나 클러스터에서 노드를 제거하기 위해**프로비저닝 해제** 하거나, 기존 격리된 노드를 실행 노드로 마이그레이션하기 위해 **iso_migrate** 입니다. default = **active** 입니다.

변수	description
node_type	**[automationcontroller]** 그룹의 경우 다음을 수행합니다. 이 그룹에 대해 두 개의 유효한 **node_type** 을 할당할 수 있습니다. **node_type=control** 은 노드가 일반 작업이 아닌 프로젝트 및 인벤토리 업데이트만 실행한다는 것을 의미합니다. **node_type=hybrid** 는 모든 것을 실행할 수 있습니다. 이 그룹의 기본값 = **hybrid** **[execution_nodes]** 그룹의 경우: 이 그룹에 대해 두 개의 유효한 **node_type** 을 할당할 수 있습니다. **node_type=hop** 은 노드가 실행 노드로 작업을 전달함을 나타냅니다. **node_type=execution** 은 노드에서 작업을 실행할 수 있음을 나타냅니다. 이 그룹의 기본값 = **실행**.
피어	*선택 사항* **peers** 변수는 특정 호스트 또는 그룹이 연결되는 노드를 나타내는 데 사용됩니다. 이 변수가 정의되면 특정 호스트 또는 그룹에 대한 아웃바운드 연결이 설정됩니다. 이 변수는 다른 노드와의 네트워크 연결을 설정하는 데 사용되는 **receptor.conf** 파일에 **tcp-peer** 항목을 추가하는 데 사용됩니다. peers 변수는 인벤토리에서 쉼표로 구분된 호스트 및 그룹 목록일 수 있습니다. 이 문제는 **receptor.conf** 파일을 구성하는 데 사용되는 호스트 세트로 해결되었습니다.
pg_database	postgreSQL 데이터베이스의 이름입니다. default = **awx**.
pg_host	postgreSQL 호스트는 외부에서 관리되는 데이터베이스일 수 있습니다.

변수	description
pg_password	postgreSQL 데이터베이스의 암호입니다. **pg_password** 에 대한 특수 문자 사용은 제한됩니다. **!**, **#**, **0** 및 **@** 문자가 지원됩니다. 다른 특수 문자를 사용하면 설정이 실패할 수 있습니다. 참고 PostgreSQL 13에서는 사용자 암호를 보다 안전하게 저장할 수 있으므로 설치 시 인벤토리 파일에 **pg_hashed_password** 를 더 이상 제공할 필요가 없습니다. 설치 프로그램의 인벤토리 파일에 **pg_password** 를 제공하는 경우 PostgreSQL은 SCRAM-SHA-256 해시를 사용하여 설치 프로세스의 일부로 해당 암호를 보호합니다.
pg_port	사용할 postgreSQL 포트입니다. 기본값 = 5432
pg_ssl_mode	사용 가능한 두 가지 모드 중 하나를 선택합니다. **prefer** 및 **verify-full**. 클라이언트 측에서 적용된 SSL의 경우 **verify-full** 로 설정합니다. default = **prefer**.
pg_username	postgreSQL 데이터베이스 사용자 이름입니다. default = **awx**.
postgres_ssl_cert	postgreSQL SSL 인증서의 위치입니다. **/path/to/pgsql_ssl.cert**
postgres_ssl_key	postgreSQL SSL 키의 위치입니다. **/path/to/pgsql_ssl.key**
postgres_use_cert	postgreSQL 사용자 인증서의 위치입니다. **/path/to/pgsql.crt**
postgres_use_key	postgreSQL 사용자 키의 위치입니다. **/path/to/pgsql.key**

변수	description
postgres_use_ssl	postgreSQL에서 SSL을 사용하는 경우 이 변수를 사용합니다.
postgres_max_connections	설치 관리자 관리 postgreSQL을 사용하는 경우 적용할 최대 데이터베이스 연결 설정입니다.Maximum database connections setting to apply if you are using installer-managed postgreSQL. 값을 선택하는 데 도움이 되는 자동화 컨트롤러 관리 가이드의 PostgreSQL 데이터베이스 구성 을 참조하십시오. VM 기반 설치의 기본값은 단일 노드의 경우 200이고 클러스터의 경우 1024입니다.
receptor_listener_port	수신기 연결에 사용할 포트입니다. Default = 27199
supervisor_start_retry_count	지정하는 경우 **startretries = <value specified** >를 supervisor 구성 파일(/etc/supervisord.d/tower.ini)에 추가합니다. **startretries** 에 대한 자세한 내용은program:x 섹션 값을 참조하십시오. 기본값이 없습니다.
web_server_ssl_cert	*선택 사항* **/path/to/webserver.cert** **automationhub_ssl_cert** 와 동일하지만 웹 서버 UI 및 API의 경우 마찬가지입니다.
web_server_ssl_key	*선택 사항* **/path/to/webserver.key** **automationhub_server_ssl_key** 와 동일하지만 웹 서버 UI 및 API의 경우

A.4. ANSIBLE 변수

다음 변수는 Ansible Automation Platform이 원격 호스트와 상호 작용하는 방법을 제어합니다.

특정 플러그인과 관련된 변수에 대한 자세한 내용은 Ansible.Builtin 설명서를 참조하십시오.

글로벌 구성 옵션 목록은 Ansible 구성 설정을 참조하십시오 .

변수	description
ansible_connection	대상 호스트에서 작업에 사용되는 연결 플러그인입니다. 이는 모든 Ansible 연결 플러그인의 이름이 될 수 있습니다. SSH 프로토콜 유형은 **smart**,**ssh** 또는 **paramiko** 입니다. default = **smart**
ansible_host	**inventory_hostname** 대신 사용할 대상 호스트의 IP 또는 이름입니다.
ansible_port	연결 포트 번호입니다. 기본값: ssh의 경우 22
ansible_user	호스트에 연결할 때 사용할 사용자 이름입니다.
ansible_password	호스트에 인증할 암호입니다. 이 변수를 일반 텍스트로 저장하지 마십시오. 항상 자격 증명 모음을 사용합니다.
ansible_ssh_private_key_file	SSH에서 사용하는 개인 키 파일입니다. 여러 키를 사용하는 경우 유용하며 SSH 에이전트를 사용하지 않습니다.
ansible_ssh_common_args	이 설정은 항상 **sftp**,**scp**, **ssh** 의 기본 명령줄에 추가됩니다. 특정 호스트 또는 그룹에 대해 ProxyCommand를 구성하는 데 유용합니다.
ansible_sftp_extra_args	이 설정은 항상 기본 **sftp** 명령줄에 추가됩니다.
ansible_scp_extra_args	이 설정은 항상 기본 **scp** 명령줄에 추가됩니다.
ansible_ssh_extra_args	이 설정은 항상 기본 **ssh** 명령줄에 추가됩니다.
ansible_ssh_pipelining	SSH 파이프링 사용 여부를 결정합니다. 이렇게 하면 **ansible.cfg** 에서 pipelining 설정을 덮어쓸 수 있습니다. SSH 키 기반 인증을 사용하는 경우 SSH 에이전트에서 키를 관리해야 합니다.
ansible_ssh_executable	버전 2.2에 추가되었습니다. 이 설정은 시스템 SSH를 사용하기 위한 기본 동작을 재정의합니다. **ansible.cfg** 에서 ssh_executable 설정을 덮어쓸 수 있습니다.

변수	description
ansible_shell_type	대상 시스템의 쉘 유형입니다. **ansible_shell_executable** 을 비 Bourne(sh) 호환 쉘로 설정하지 않으면 이 설정을 사용하지 마십시오. 기본적으로 명령은 sh-style 구문을 사용하여 포맷됩니다. 이를 **csh** 또는 **fish** 로 설정하면 대상 시스템에서 명령이 대신 해당 쉘의 구문을 따릅니다.
ansible_shell_executable	이렇게 하면 Ansible 컨트롤러가 대상 시스템에서 사용하는 쉘을 설정하고 기본값은 **/bin/sh** 인 **ansible.cfg** 에서 실행 파일을 덮어씁니다. **/bin/sh** 가 대상 시스템에 설치되지 않았거나 sudo에서 실행할 수 없는 경우가 아니면 이 변수를 변경하지 마십시오.
inventory_hostname	이 변수는 인벤토리 스크립트 또는 Ansible 구성 파일에서 시스템의 호스트 이름을 가져옵니다. 이 변수의 값을 설정할 수 없습니다. 이 값은 구성 파일에서 가져 오기 때문에 실제 런타임 호스트 이름 값은 이 변수에서 반환하는 내용과 다를 수 있습니다.

A.5. 이벤트 기반 ANSIBLE 컨트롤러 변수

변수	description
automationedacontroller_admin_password	이벤트 기반 Ansible 컨트롤러 인스턴스에서 사용하는 관리자 암호입니다. 인벤토리 파일에 일반 텍스트로 제공되면 암호를 따옴표로 묶어야 합니다.
automationedacontroller_admin_username	Django에서 Event-Driven Ansible 컨트롤러에서 admin 슈퍼유저를 식별하고 생성하는 데 사용하는 사용자 이름입니다. default = **admin**
automationedacontroller_admin_email	이벤트 기반 Ansible 컨트롤러에 대해 admin 사용자에 대해 Django에서 사용하는 이메일 주소입니다. Default = **admin@example.com**
automationedacontroller_allowed_hostnames	이벤트 기반 Ansible 컨트롤러에 대한 사용자 액세스에 사용할 추가 주소 목록입니다. default = 빈 목록

변수	description
automationedacontroller_controller_verify_ssl	이벤트 기반 Ansible 컨트롤러에서 호출할 때 자동화 컨트롤러의 웹 인증서를 확인하는 데 사용되는 부울 플래그입니다. 검증은 **true** 입니다. 검증되지 않은 것은 **false** 입니다. default = **false**
automationedacontroller_disable_https	HTTPS 이벤트 기반 Ansible 컨트롤러를 비활성화하는 부울 플래그입니다. default = **false**
automationedacontroller_disable_hsts	HSTS 이벤트 기반 Ansible 컨트롤러를 비활성화하는 부울 플래그입니다. default = **false**
automationedacontroller_gunicorn_workers	gunicorn을 통해 제공되는 API의 작업자 수입니다. default = (# of cores or threads) * 2 + 1
automationedacontroller_max_running_activations	노드당 동시에 실행되는 최대 활성화 수입니다. 0보다 커야 하는 정수입니다. 기본값 = 12
automationedacontroller_nginx_tls_files_remote	인증서 소스가 원격 호스트(true) 또는 로컬(false)에 있는지 여부를 지정하는 부울 플래그입니다. default = **false**
automationedacontroller_pg_database	이벤트 기반 Ansible 컨트롤러에서 사용하는 Postgres 데이터베이스입니다. default = **automtionedacontroller**.
automationnedacontroller_pg_host	외부에서 관리되는 데이터베이스일 수 있는 이벤트 기반 Ansible 컨트롤러에서 사용하는 Postgres 데이터베이스의 호스트 이름입니다.
automationedacontroller_pg_password	Event-Driven Ansible 컨트롤러에서 사용하는 Postgres 데이터베이스의 암호입니다. Automation **edacontroller_pg_password** 에는 특수 문자 사용이 제한됩니다. **!**, **#**, **0** 및 **@** 문자가 지원됩니다. 다른 특수 문자를 사용하면 설정이 실패할 수 있습니다.

변수	description
automationedacontroller_pg_port	Event-Driven Ansible 컨트롤러에서 사용하는 Postgres 데이터베이스의 포트 번호입니다. 기본값 = **5432**.
automationedacontroller_pg_username	Event-Driven Ansible 컨트롤러 Postgres 데이터베이스의 사용자 이름입니다. default = **automationedacontroller**.
automationedacontroller_rq_workers	이벤트 기반 Ansible 컨트롤러에서 사용하는 Redis Queue(RQ) 작업자 수입니다. RQ 작업자는 백그라운드에서 실행되는 Python 프로세스입니다. default = (# of cores or threads) * 2 + 1
automationedacontroller_ssl_cert	*선택 사항* **/root/ssl_certs/eda.*<example>*.com.crt** Automation **hub_ssl_cert** 와 동일하지만 이벤트 기반 Ansible 컨트롤러 UI 및 API용입니다.
automationedacontroller_ssl_key	*선택 사항* **/root/ssl_certs/eda.*<example>*.com.key** Automation **hub_server_ssl_key** 와 동일하지만 이벤트 기반 Ansible 컨트롤러 UI 및 API용입니다.
automationedacontroller_user_headers	이벤트 기반 Ansible 컨트롤러의 nginx 구성에 추가할 추가 nginx 헤더 목록입니다. default = 빈 목록