Red Hat Ansible Automation Platform 2.2

Red Hat Ansible Automation Platform 설치 가이드

이 가이드에서는 Red Hat Ansible Automation Platform에서 지원되는 설치 시나리오에 대한 절차 및 참조 정보를 제공합니다.

Last Updated: 2023-12-28

Red Hat Ansible Automation Platform 2.2 Red Hat Ansible Automation Platform 설치 가이드

이 가이드에서는 Red Hat Ansible Automation Platform에서 지원되는 설치 시나리오에 대한 절차 및 참조 정보를 제공합니다.

법적 공지

초록

피드백 제공: 이 문서를 개선하거나 오류를 찾을 수 있는 제안이 있는 경우 문서 구성 요소를 사용하여 Ansible Automation Platform Jira 프로젝트에 문제를 생성하기 위해 에서 기술 지원에 문의하십시오.

차례

머리말

Red Hat Ansible Automation Platform에 관심을 가져 주셔서 감사합니다. Ansible Automation Platform은 팀이 Ansible 기반 환경에 제어, 지식 및 위임을 추가하여 복잡한 다단계 배포를 관리할 수 있도록 지원하는 상용 서비스입니다.

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

보다 포괄적 수용을 위한 오픈 소스 용어 교체

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

1장. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 계획

Red Hat Ansible Automation Platform은 Red Hat Enterprise Linux 및 Red Hat Openshift 모두에서 지원됩니다. 이 가이드를 사용하여 Red Hat Enterprise Linux에 Red Hat Ansible Automation Platform 설치를 계획하십시오.

Red Hat OpenShift Container Platform 환경에 Red Hat Ansible Automation Platform을 설치하려면 OpenShift Container Platform 에 Red Hat Ansible Automation Platform Operator 배포를 참조하십시오.

1.1. RED HAT ANSIBLE AUTOMATION PLATFORM 시스템 요구 사항

Red Hat Ansible Automation Platform 설치 계획 및 사용 사례에 맞는 자동화 메시 토폴로지 설계 시 이 정보를 사용합니다.

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

표 1.1. 기본 시스템

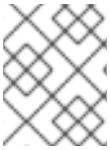
	필수 항목	참고
서브스크립션	유효한 Red Hat Ansible Automation Platform	
OS	Red Hat Enterprise Linux 8.4 이상 64비트(x86)	Red Hat Ansible Automation Platform은 OpenShift에서 지원되며 자세한 내용은 OpenShift Container Platform에서 Red Hat Ansible Automation Platform Operator 배포를 참조하십시오.
Ansible	버전 2.2 필요	Ansible이 시스템에 없는 경우 설정 플레이북은 **ansible-core** 2.13을 설치합니다.
Python	3.8 이상	

다음은 프로젝트 업데이트 및 컬렉션으로 작업하는 데 필요합니다.

- 다음 도메인 이름이 방화벽 또는 프록시의 성공적인 연결을 위한 허용 목록에 포함되어 있는지 확인하고 자동화 허브 또는 Galaxy 서버에서 컬렉션을 다운로드하십시오.
  - **galaxy.ansible.com**
  - **cloud.redhat.com**
  - **console.redhat.com**
  - **sso.redhat.com**
- 자체 서명된 인증서를 사용하거나 Red Hat 도메인에 대해 SSL 검사를 비활성화해야 합니다.

1.1.1. 자동화 컨트롤러

자동화 컨트롤러는 분산 시스템입니다. 여기서 다양한 소프트웨어 구성 요소를 여러 컴퓨팅 노드에 공동 배치하거나 배포할 수 있습니다. 설치 프로그램에서 노드 유형의 제어, 하이브리드, 실행 및 홉은 사용자가 사용 사례에 적합한 토폴로지를 설계할 수 있도록 추상화로 제공됩니다. 다음 표에서는 노드 크기 조정을 위한 권장 사항을 제공합니다.



참고

홉 노드를 제외한 모든 노드에서 실행 환경 저장을 위해 최소 20GB를 **/var/lib/awx** 에 할당합니다.

실행 노드	필수 항목	참고
RAM	16GB	
CPU	4	• 자동화를 실행합니다. 더 많은 포크 실행을 위한 용량을 늘리기 위해 메모리 및 CPU를 늘리십시오.
제어 노드	필수 항목	참고
RAM	16GB	
CPU	4	• 프로젝트 업데이트 및 정리 작업을 포함하여 이벤트를 처리하고 클러스터 작업을 실행합니다. CPU 및 메모리를 늘리면 작업 이벤트 처리에 도움이 될 수 있습니다.
하이브리드 노드	필수 항목	참고
RAM	16GB	
CPU	4	• 자동화 및 클러스터 작업을 모두 실행합니다. 실행 및 제어 노드에 대한 주석은 이 노드 유형에 적용됩니다.
노드 홉	필수 항목	참고
RAM	16GB	

CPU	4	• Automation Mesh의 한 부분에서 다른 부분으로 트래픽을 라우팅하는 역할을 합니다(예: bastion 호스트가 다른 네트워크로 될 수 있음). RAM은 처리량에 영향을 미칠 수 있으며 CPU 활동이 낮습니다. 네트워크 대역폭 및 대기 시간은 일반적으로 RAM/CPU보다 더 중요한 요소입니다.
disk: 서비스 노드	40GB 전용 하드 디스크 공간	• **자동화 컨트롤러**: 파일 및 작업 디렉터리 저장을 위해 최소 20GB를 **/var/** 에 지정합니다. • 스토리지 볼륨은 1500 IOPS의 최소 기준치로 평가되어야 합니다. • 프로젝트는 제어 및 하이브리드, 작업 기간 동안 실행 노드에도 저장됩니다. 클러스터에 많은 대규모 프로젝트가 있는 경우 디스크 공간 오류를 방지하려면 /var/lib/awx/projects에 GB를 두 번 사용하는 것이 좋습니다.
disk: 데이터베이스 노드	20GB 전용 하드 디스크 공간	• 150GB 이상 권장 • 스토리지 볼륨은 높은 기준 IOPS (1500 이상)에 대해 평가되어야 합니다.
브라우저	현재 지원되는 Mozilla VolumeSnapshotFox 또는 Google Chrome	
데이터베이스	PostgreSQL 버전 13	

추가 리소스

- 자동화 컨트롤러 사용을 인증하려면 서브스크립션 가져오기를 참조하십시오.

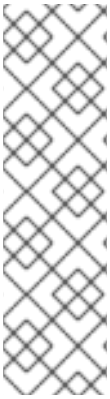
1.1.2. Automation hub

Automation hub를 사용하면 Red Hat Ansible 및 Certified Partners에서 인증된 새로운 자동화 컨텐츠를 검색하고 사용할 수 있습니다. Ansible 자동화 허브에서는 클라우드 자동화, 네트워크 자동화 및 보안 자동

화와 같은 사용 사례를 위해 Red Hat 및 파트너사에서 개발한 자동화 콘텐츠인 Ansible Collections를 검색하고 관리할 수 있습니다.

Automation Hub에는 다음과 같은 시스템 요구 사항이 있습니다.

	필수 항목	참고
RAM	8GB 최소	• 8GB RAM (최소 DestinationRule 평가판 설치 시 권장) • 8GB RAM (외부 독립 실행형 PostgreSQL 데이터베이스의 경우 최소) • 구성의 포크 기반 용량의 경우 추가 리소스를 참조하십시오.
CPU	2개 최소	• 구성의 포크 기반 용량의 경우 추가 리소스를 참조하십시오.
disk: 서비스 노드	60GB 전용 하드 디스크 공간	• 스토리지 볼륨은 1500 IOPS의 최소 기준치로 평가되어야 합니다.
disk: 데이터베이스 노드	20GB 전용 하드 디스크 공간	• 150GB 이상 권장 • 스토리지 볼륨은 높은 기준 IOPS (1500 이상)에 대해 평가되어야 합니다.
브라우저	현재 지원되는 Mozilla VolumeSnapshotFox 또는 Google Chrome	
데이터베이스	PostgreSQL 버전 13	



참고

- 모든 자동화 컨트롤러 데이터는 데이터베이스에 저장됩니다. 데이터베이스 스토리지는 관리되는 호스트 수, 작업 실행 수, 팩트 캐시에 저장된 팩트 수 및 개별 작업의 작업 수와 함께 증가합니다. 예를 들어, 플레이북은 250일에 한 시간(하루 24회) 실행되는 호스트에서 20개의 작업이 매주 데이터베이스에 800000 이상의 이벤트를 저장합니다.
- 데이터베이스에 공간이 충분하지 않은 경우 오래된 작업이 실행되고 정기적으로 팩트를 정리해야 합니다. 자세한 내용은 *자동화 컨트롤러 관리 가이드의 관리* 작업에서 참조하십시오.

Amazon EC2

- m5.large 이상의 인스턴스 크기
- 호스트가 100개 이상인 경우 m4.xlarge의 인스턴스 크기

Red Hat Ansible Automation Platform 요구 사항에 대한 추가 정보

- 실제 RAM 요구 사항은 동시에 관리할 호스트 자동화 컨트롤러 수(작업 템플릿의 **forks** 매개변수 또는 시스템 **ansible.cfg** 파일에서 제어함)에 따라 다릅니다. 리소스 충돌을 방지하기 위해 Ansible은 자동화 컨트롤러를 위한 10개 포크당 1GB의 메모리 + 자동화 컨트롤러에 대한 예약 2GB를 권장합니다. 자세한 내용은 Automation Controller Capacity Determination and Job Impact 를 참조하십시오. **포크** 가 400으로 설정된 경우 42GB의 메모리가 권장됩니다.
- 포크 수가 총 호스트 수보다 작으면 호스트에 걸쳐 더 많은 통과가 필요합니다. 롤링 업데이트를 사용하거나 자동화 컨트롤러에 빌드된 프로비저닝 콜백 시스템을 사용할 때 이러한 RAM 제한을 방지할 수 있습니다. 여기서 각 시스템이 대기열에 입력되고 가능한 한 빨리 처리되거나 자동화 컨트롤러에서 AMI와 같은 이미지를 생성하거나 배포하는 경우입니다. 이 모든 것은 더 큰 환경을 관리하기 위한 좋은 접근 방식입니다. 자세한 내용은 https://access.redhat.com/ 에서 Red Hat 고객 포털을 통해 Ansible 지원에 문의하십시오.
- Ansible Automation Platform에서 관리하는 시스템의 요구 사항은 Ansible의 요구 사항과 동일합니다. Ansible *사용자 가이드에서 시작하기 를 참조하십시오*.

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 인스턴스에서 데이터베이스에 액세스할 수 있는지 확인하기 위해 **awx-manage check_db** 명령을 사용하여 이를 수행할 수 있습니다.

PostgreSQL 구성

선택적으로 PostgreSQL 데이터베이스를 Red Hat Ansible Automation Platform 설치 프로그램에서 관리하지 않는 별도의 노드로 구성할 수 있습니다. Ansible Automation Platform 설치 프로그램이 데이터베이스 서버를 관리하면 일반적으로 대부분의 워크로드에 권장되는 기본값을 사용하여 서버를 구성합니다. 그러나 **ansible_memtotal_mb** 이 데이터베이스 서버의 총 메모리 크기인 독립 실행형 데이터베이스 서버 노드에 대해 이러한 PostgreSQL 설정을 조정할 수 있습니다.

```
max_connections == 1024
shared_buffers == ansible_memtotal_mb*0.3
```

```
work_mem == ansible_memtotal_mb*0.03
maintenance_work_mem == ansible_memtotal_mb*0.04
```

PostgreSQL 서버 튜닝에 대한 자세한 내용은 PostgreSQL 설명서를 참조하십시오 .

Red Hat Ansible Automation Platform은 Ansible 플레이북에 따라 달라지며 자동화 컨트롤러를 설치하기 전에 안정적인 최신 버전의 Ansible을 설치해야 하지만 Ansible 수동 설치가 더 이상 필요하지 않습니다.

새로운 설치 시 자동화 컨트롤러는 Ansible 2.2의 최신 릴리스 패키지를 설치합니다.

번들로 번들 Ansible Automation Platform 설치를 수행하는 경우 설치 프로그램은 번들에서 Ansible(및 종속 항목)을 설치하려고 합니다.

자체적으로 Ansible을 설치하도록 선택하는 경우 Ansible Automation Platform 설치 프로그램은 Ansible이 설치되었음을 감지하여 다시 설치하지 않습니다. **yum** 과 같은 패키지 관리자를 사용하여 Ansible을 설치해야 하며 Red Hat Ansible Automation Platform이 제대로 작동하려면 안정적인 최신 버전을 설치해야 합니다. Ansible 버전 2.9는 |at| 버전 3.8 이상에 필요합니다.

- 자체적으로 Ansible을 설치하도록 선택하는 경우 Ansible Automation Platform 설치 프로그램은 Ansible이 설치되었음을 탐지하고 다시 설치하지 않습니다.

참고

yum 과 같은 패키지 관리자를 사용하여 Ansible을 설치해야 하며 Red Hat Ansible Automation Platform이 제대로 작동하려면 패키지 관리자의 최신 버전을 설치해야 합니다. Ansible 버전 2.9는 버전 3.8 이상에 필요합니다.

1.2. 네트워크 포트 및 프로토콜

Red Hat Ansible Automation Platform (AAP)은 여러 포트를 사용하여 서비스와 통신합니다. 이러한 포트가 작동하려면 Red Hat Ansible Automation Platform 서버로 들어오는 연결에 열려 있어야 합니다. 이러한 포트를 사용할 수 있으며 서버 방화벽에 의해 차단되지 않았는지 확인합니다.

다음 표에서는 각 애플리케이션에 필요한 기본 Red Hat Ansible Automation Platform 대상 포트를 제공합니다.

참고

아래에 나열된 기본 대상 포트 및 설치 프로그램 인벤토리는 구성 가능합니다. 환경에 맞게 구성하도록 선택하는 경우 동작 변경이 발생할 수 있습니다.

표 1.2. PostgreSQL

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치하는 동안 원격 액세스

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
5432	TCP	Postgres	인바운드 및 아웃 바운드	**pg_port**	기본 포트 컨트롤러에서 데이터베이스 포트로의 ALLOW 연결

표 1.3. 자동화 컨트롤러

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
80	TCP	HTTP	인바운드	**nginx_http_port**	UI/API
443	TCP	HTTPS	인바운드	**nginx_https_port**	UI/API
5432	TCP	PostgreSQL	인바운드 및 아웃 바운드	**pg_port**	내부 데이터베이스가 다른 구성 요소와 함께 사용되는 **경우에만** 엽니다. 그렇지 않으면 이 포트를 열 수 없습니다. 클러스터의 하이브리드 모드
27199	TCP	수용체	인바운드 및 아웃 바운드	**receptor_listener_port**	필수 및 자동 컨트롤 플레인 클러스터링을 위해 모든 컨트롤러에서 ALLOW 리스너 포트를 수용

표 1.4. 노드 홉핑

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
27199	TCP	수용체	인바운드 및 아웃 바운드	**receptor_listener_port**	메시 Controller(s)에서 수신 포트로의 ALLOW 연결

표 1.5. 실행 노드

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
27199	TCP	수용체	인바운드 및 아웃 바운드	**receptor_listener_port**	Mesh - 컨트롤러와 직접 피어링된 노드입니다. 홉 노드와 관련이 없습니다. 27199는 실행 노드의 양방향입니다. Controller(s)에서 Receptor 포트(non-ECDHE connected nodes)까지 ALLOW 연결 Hop node(s)에서 Receptor 포트(홉 노드를 통해 중계되는 경우)로의 ALLOW 연결

표 1.6. 제어 노드

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
27199	TCP	수용체	인바운드 및 아웃 바운드	**receptor_listener_port**	Mesh - 컨트롤러와 직접 피어링된 노드입니다. 관련된 직접 노드 27199는 실행 노드의 양방향입니다. controller(s)에서 Receptor port for non-ECDHE connected nodes에 대한 ENABLE 연결 홉 노드를 통해 중계된 경우 hop node에서 Receptor 포트로의 ENABLE 연결
443	TCP	Podman	인바운드	**nginx_https_port**	UI/API

표 1.7. 하이브리드 노드

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
27199	TCP	수용체	인바운드 및 아웃 바운드	**receptor_listener_port**	Mesh - 컨트롤러와 직접 피어링된 노드입니다. 홉 노드와 관련이 없습니다. 27199는 실행 노드의 양방향입니다. controller(s)에서 Receptor port for non-ECDHE connected nodes에 대한 ENABLE 연결 홉 노드를 통해 중계된 경우 hop node에서 Receptor 포트로의 ENABLE 연결
443	TCP	Podman	인바운드	**nginx_https_port**	UI/API

표 1.8. Automation hub

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
80	TCP	HTTP	인바운드	고정 값	사용자 인터페이스
443	TCP	HTTPS	인바운드	고정 값	사용자 인터페이스

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
5432	TCP	PostgreSQL	인바운드 및 아웃 바운드	**automationhub_pg_port**	내부 데이터베이스가 다른 구성 요소와 함께 사용되는 **경우에만** 엽니다. 그렇지 않으면 이 포트를 열 수 없습니다.

표 1.9. 서비스 카탈로그

포트	프로토콜	Service	direction	설치 프로그램 인벤토리 변수	필요한 항목
22	TCP	SSH	인바운드 및 아웃 바운드	**ansible_port**	설치
443	TCP	HTTPS	인바운드	**nginx_https_port**	Service Catalog 사용자 인터페이스에 액세스
5432	TCP	PostgreSQL	인바운드 및 아웃 바운드	**pg_port**	내부 데이터베이스가 사용되는 **경우에만** 엽니다. 그렇지 않으면 이 포트를 열 수 없습니다.

표 1.10. Red Hat Insights for Red Hat Ansible Automation Platform

URL	필요한 항목
http://api.access.redhat.com:443	일반 계정 서비스, 서브스크립션
https://cert-api.access.redhat.com:443	Insights 데이터 업로드
https://cert.cloud.redhat.com:443	인벤토리 업로드 및 Cloud Connector 연결
https://cloud.redhat.com	Insights 대시보드에 액세스

표 1.11. Automation Hub

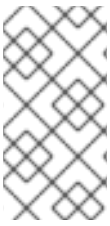
URL	필요한 항목
https://console.redhat.com:443	일반 계정 서비스, 서브스크립션
https://sso.redhat.com:443	TCP
https://automation-hub-prd.s3.amazonaws.com	
https://galaxy.ansible.com	Ansible Community 큐레이팅 Ansible 콘텐츠
https://ansible-galaxy.s3.amazonaws.com	
https://registry.redhat.io:443	Red Hat 및 파트너에서 제공하는 컨테이너 이미지에 액세스
https://cert.cloud.redhat.com:443	Red Hat 및 파트너 큐레이션 Ansible Collections

표 1.12. 실행 환경(EE)

URL	필요한 항목
https://registry.redhat.io:443	Red Hat 및 파트너에서 제공하는 컨테이너 이미지에 액세스

1.3. RED HAT ANSIBLE AUTOMATION PLATFORM 서브스크립션 연결

Red Hat Ansible Automation Platform을 설치하기 전에 모든 노드에 유효한 서브스크립션이 **있어야 합니다**. Ansible Automation Platform 서브스크립션을 연결하면 설치를 진행하는 데 필요한 하위 전용 리소스에 액세스할 수 있습니다.



참고

Red Hat 계정에서 Simple Content Access Mode 를 활성화한 경우 서브스크립션을 연결할 필요가 없습니다. 활성화하면 Ansible Automation Platform을 설치하기 전에 시스템을 RHSM(Red Hat Subscription Management) 또는 Satellite에 등록해야 합니다. 자세한 내용은 Simple Content Access Mode 를 참조하십시오.

절차

1. Red Hat Ansible Automation Platform 서브스크립션의 **pool_id** 를 가져옵니다.

```
# subscription-manager list --available --all | grep "Ansible Automation Platform" -B 3 -A 6
```

예제

subsciption-manager list 명령의 출력 예. **Pool ID:** 섹션에 표시된 **pool_id** 를 가져옵니다.

```
Subscription Name: Red Hat Ansible Automation, Premium (5000 Managed Nodes)
  Provides: Red Hat Ansible Engine
```

```
Red Hat Ansible Automation Platform
SKU: MCT3695
Contract: ````
Pool ID: <pool_id>
Provides Management: No
Available: 4999
Suggested: 1
```

2. 서브스크립션을 연결합니다.

```
# subscription-manager attach --pool=<pool_id>
```

이제 Red Hat Ansible Automation Platform 서브스크립션을 모든 노드에 연결했습니다.

검증

- 서브스크립션이 성공적으로 연결되었는지 확인합니다.

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

문제 해결

- Ansible Automation Platform 설치 프로그램과 함께 제공된 특정 패키지를 찾을 수 없거나 ***구성 메시지로 비활성화된 리포지토리*** 가 표시되면 명령을 사용하여 리포지토리를 활성화하십시오.
  Red Hat Ansible Automation Platform 2.2 for RHEL 8

  ```
  subscription-manager repos --enable ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms
  ```

  Red Hat Ansible Automation Platform 2.2 for RHEL 9

  ```
  subscription-manager repos --enable ansible-automation-platform-2.2-for-rhel-9-x86_64-rpms
  ```

1.4. RED HAT ANSIBLE AUTOMATION PLATFORM 플랫폼 구성 요소

Red Hat Ansible Automation Platform은 다음과 같은 구성 요소로 구성됩니다.

Ansible 자동화 허브

Ansible Content Collections의 인증된 콘텐츠에 대한 리포지토리입니다. Ansible 자동화 허브는 Red Hat과 파트너사가 콘텐츠를 게시하고 고객이 지원되는 인증된 Ansible Content Collections를 검색할 수 있는 중앙 집중식 리포지토리입니다. Red Hat Ansible Certified Content는 Red Hat에서 테스트하고 지원하는 컨텐츠를 사용자에게 제공합니다.

프라이빗 자동화 허브

프라이빗 자동화 허브는 콘텐츠를 동기화하기 위해 연결이 끊긴 온프레미스 솔루션 및 온프레미스 솔루션을 모두 제공합니다. Red Hat 클라우드 자동화 허브의 컬렉션 및 실행 환경 이미지를 동기화하여 사용자 지정 자동화 컬렉션 및 실행 이미지를 저장하고 제공할 수 있습니다. Ansible Galaxy 또는 기타 컨테이너 레지스트리와 같은 다른 소스를 사용하여 개인 자동화 허브에 콘텐츠를 제공할 수도 있습니다. 프라이빗 자동화 허브는 엔터프라이즈 디렉터리 및 CI/CD 파이프라인에 통합할 수 있습니다.

자동화 컨트롤러

UI(사용자 인터페이스) 및 RESTful API(애플리케이션 프로그래밍 인터페이스)를 사용하여 Ansible 자동화를 제어, 보안 및 관리하기 위한 엔터프라이즈 프레임워크입니다.

자동화 서비스 카탈로그

Automation 서비스 카탈로그는 Red Hat Ansible Automation Platform 내의 서비스입니다. 자동화 서비스 카탈로그를 사용하면 다양한 환경에서 Ansible 자동화 컨트롤러에서 제품 카탈로그 소스를 구성하고 관리할 수 있습니다.

자동화 서비스 카탈로그를 사용하면 다음을 수행할 수 있습니다.

- 개별 플랫폼 인벤토리에 멀티 수준 승인을 적용합니다.
- 콘텐츠를 플랫폼에서 포트폴리오로 제품 형식으로 구성합니다.
- 특정 사용자 그룹과 공유할 포트폴리오를 선택합니다.
- 사용자 요청 실행을 유도하는 값에 대한 경계를 설정합니다.

자동화 메시

자동화 메시는 기존 네트워크를 사용하여 서로 피어 간 연결을 설정하는 노드를 통해 대규모 및 분산된 작업자 컬렉션에 걸쳐 작업을 쉽게 배포하기 위한 오버레이 네트워크입니다.

자동화 메시는 다음을 제공합니다.

- 개별적으로 확장되는 동적 클러스터 용량을 통해 다운 타임을 최소화하여 노드를 생성, 등록, 그룹, ungroup 및 deregister할 수 있습니다.
- 컨트롤 플레인 용량과 독립적으로 플레이북 실행 용량을 확장할 수 있는 컨트롤 및 실행 플레인 분리.
- 대기 시간에 탄력적이고 중단 없이 재구성할 수 있는 배포 선택 사항 및 중단 시 다른 경로를 선택하도록 동적으로 다시 라우팅합니다.
- 메시 라우팅 변경
- IPS(Federal Information Processing Standards) 준수 가능성이 있는 양방향, 다중 보호 메시 통신 가능성이 포함된 연결입니다.

자동화 실행 환경

Ansible 실행 엔진과 수백 개의 모듈이 포함된 솔루션으로 사용자가 IT 환경 및 프로세스의 모든 측면을 자동화할 수 있습니다. 실행 환경은 일반적으로 사용되는 운영 체제, 인프라 플랫폼, 네트워크 장치 및 클라우드를 자동화합니다.

Ansible Galaxy

Ansible 콘텐츠를 검색, 재사용 및 공유하기 위한 허브입니다. 커뮤니티 제공 Galaxy 콘텐츠는 사전 패키징된 역할로 자동화 프로젝트를 시작하는 데 도움이 될 수 있습니다. 인프라 프로비저닝, 애플리케이션 배포 및 기타 작업 완료에 대한 역할은 Ansible 플레이북으로 삭제되고 고객 환경에 즉시 적용할 수 있습니다.

자동화 콘텐츠 navigator

기본 명령줄 *인터페이스가 되는 텍스트 인터페이스* (TUI)로, 콘텐츠 빌드의 사용 사례, 실행 환경에서 자동화 실행, Ansible Automation Platform에서 자동화 실행, 향후 *통합 개발* 환경(IDE)의 기반을 제공하는 텍스트입니다.

1.5. RED HAT ANSIBLE AUTOMATION PLATFORM 설치 프로그램 선택 및 받기

Red Hat Enterprise Linux 환경 인터넷 연결을 기반으로 필요한 Red Hat Ansible Automation Platform 설치 관리자를 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 프로그램 다운로드에 액세스하려면 유효한 Red Hat 고객 계정이 필요합니다.

인터넷 액세스를 사용하여 설치

Red Hat Enterprise Linux 환경이 인터넷에 연결된 경우 Red Hat Ansible Automation Platform (AAP) 설치 프로그램을 선택합니다. 인터넷 액세스를 사용하여 설치하면 필요한 최신 리포지토리, 패키지 및 종속 항목을 검색합니다. 다음 방법 중 하나를 선택하여 AAP 설치 프로그램을 설정합니다.

tarball 설치

1. https://access.redhat.com/downloads/content/480로 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
   x86_64용 RHEL 8용 V.2.2

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.2-for-rhel-8-x86_64-rpms
ansible-automation-platform-installer
```

   x86-64용 RHEL 9용 V.2.2

```
$ sudo dnf install --enablerepo=ansible-automation-platform-2.2-for-rhel-9-x86_64-rpms
ansible-automation-platform-installer
```

참고

DNF 설치 에서는 리포지토리가 기본적으로 비활성화되어 있으므로 리포지토리를 활성화합니다.

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

인터넷 액세스없이 설치

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

1. https://access.redhat.com/downloads/content/480로 이동합니다.
2. **Ansible Automation Platform <latest-version> 설치 번들에 대해지금 다운로드를 클릭합니다**.
3. 파일을 추출합니다.

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

1.6. 설치 프로그램 인벤토리 파일 정보

Red Hat Ansible Automation Platform은 인벤토리 파일을 사용하여 논리적으로 구성된 인프라의 관리 노드 또는 호스트 목록에 대해 작동합니다. Red Hat Ansible Automation Platform 설치 프로그램 인벤토리 파일을 사용하여 설치 시나리오를 지정하고 Ansible에 대한 호스트 배포를 설명할 수 있습니다. Ansible은 인벤토리 파일을 사용하여 단일 명령으로 다수의 호스트를 관리할 수 있습니다. 또한 인벤토리를 사용하면 지정해야 하는 명령줄 옵션의 수를 줄임으로써 Ansible을 보다 효율적으로 사용할 수 있습니다.

인벤토리 파일은 보유 중인 인벤토리 플러그인에 따라 여러 형식 중 하나일 수 있습니다. 가장 일반적인 형식은 **INI** 및 **YAML** 입니다. 이 문서에 나열된 인벤토리 파일은 INI 형식으로 표시됩니다.

인벤토리 파일의 위치는 사용한 설치 프로그램에 따라 다릅니다. 다음 표에서는 가능한 위치를 보여줍니다.

설치 프로그램	위치
bundle tar	**/ansible-automation-platform-setup-bundle-<latest-version>**
번들 이외의 tar	**/ansible-automation-platform-setup-<latest-version>**
RPM	**/opt/ansible-automation-platform/installer**

다음 명령을 사용하여 인벤토리에서 호스트를 확인할 수 있습니다.

```
ansible all -i <path-to-inventory-file. --list-hosts
```

인벤토리 파일 예

```
[automationcontroller]
host1.example.com
host2.example.com
Host4.example.com

[automationhub]
host3.example.com

[database]
```

```
Host5.example.com

[all:vars]
admin_password='<password>'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'

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

인벤토리 파일의 첫 번째 부분에서는 Ansible이 사용할 수 있는 호스트 또는 그룹을 지정합니다.

1.6.1. 호스트 및 그룹에 대한 지침

데이터베이스

- 외부 데이터베이스를 사용하는 경우 인벤토리 파일의 **[database]** 섹션이 올바르게 설정되어 있는지 확인합니다.
- 성능을 향상하려면 동일한 서버에서 데이터베이스 및 자동화 컨트롤러를 공동 배치하지 마십시오.

Automation hub

- **[automationhub]** 그룹에 Ansible 자동화 허브 정보 추가
- 동일한 노드에 Ansible 자동화 허브 및 자동화 컨트롤러를 설치하지 마십시오.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소 또는 정규화된 도메인 이름(FDQN)을 제공하여 사용자가 다른 노드의 Ansible 자동화 허브에서 콘텐츠를 동기화하고 설치할 수 있는지 확인합니다. **localhost** 를 사용하지 마십시오.

중요

[database] 그룹이 둘 다 동시에 설치된 경우 둘을 구분하지 않으므로 자동화 컨트롤러와 Ansible 자동화 허브 설치를 분리해야 합니다.

[database] 및 자동화 컨트롤러와 Ansible 자동화 허브에서 하나의 값을 사용하는 경우 동일한 데이터베이스를 사용합니다.

자동화 컨트롤러

- 자동화 컨트롤러는 사용하는 데이터베이스에 대한 복제 또는 장애 조치를 구성하지 않습니다. 자동화 컨트롤러는 사용자가 보유한 복제에서 작동합니다.

클러스터형 설치

- 기존 클러스터를 업그레이드할 때 기존 인스턴스 또는 인스턴스 그룹을 생략하도록 클러스터를

재구성할 수도 있습니다. 인벤토리 파일에서 인스턴스 또는 인스턴스 그룹을 생략하는 것만으로는 클러스터에서 해당 인스턴스를 제거하기에 충분하지 않습니다. 인벤토리 파일에서 인스턴스 또는 인스턴스 그룹을 생략하는 것 외에도 업그레이드를 시작하기 전에 인스턴스 또는 인스턴스 그룹도 프로비저닝 해제해야 합니다. 노드 또는 그룹 프로비저닝 을 참조하십시오. 그렇지 않으면 인스턴스 또는 인스턴스 그룹이 계속 클러스터와 통신하므로 업그레이드 중에 자동화 컨트롤러 서비스에 문제가 발생할 수 있습니다.

- 클러스터형 설치 설정을 생성하는 경우 **[localhost]** 를 모든 인스턴스의 호스트 이름 또는 IP 주소로 교체해야 합니다. 자동화 컨트롤러, 자동화 허브 및 자동화 서비스 카탈로그의 설치 프로그램은 **[localhost]** 모든 노드 및 인스턴스를 이 호스트 이름 또는 주소를 사용하여 다른 노드에 연결할 수 있어야 합니다. 즉 노드 중 하나에서 localhost **ansible_connection=local** 을 사용할 수 없습니다. 모든 노드의 호스트 이름에 동일한 형식을 사용합니다.
따라서 이 작업이 작동하지 않습니다.

```
[automationhub]
localhost ansible_connection=local
hostA
hostB.example.com
172.27.0.4
```

대신 다음 형식을 사용하십시오.

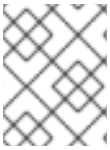
```
[automationhub]
hostA
hostB
hostC
```

또는

```
[automationhub]
hostA.example.com
hostB.example.com
hostC.example.com
```

1.6.2. 노드 또는 그룹 프로비저닝 해제

Ansible Automation Platform 설치 프로그램을 사용하여 노드 및 인스턴스 그룹을 프로비저닝 해제할 수 있습니다. 설치 프로그램을 실행하면 구성 파일과 그룹의 노드에 연결된 로그가 모두 제거됩니다.



참고

[automationcontroller] 그룹에 지정된 첫 번째 호스트를 제외하고 인벤토리의 호스트 프로비저닝을 해제할 수 있습니다.

노드를 프로비저닝 해제하려면 **node_state=deprovision** 을 인벤토리 파일 내의 노드 또는 그룹에 추가합니다.

예를 들면 다음과 같습니다.

배포에서 단일 노드를 제거하려면 다음을 수행합니다.

```
[automationcontroller]
host1.example.com
```

```
host2.example.com
host4.example.com   node_state=deprovision
```

또는

배포에서 전체 인스턴스 그룹을 제거하려면 다음을 수행합니다.

```
[instance_group_restrictedzone]
host4.example.com
host5.example.com

[instance_group_restrictedzone:vars]
node_state=deprovision
```

1.6.3. 인벤토리 변수

예제 인벤토리 파일의 두 번째 부분 **([all:vars]**)은 설치 프로그램에서 사용하는 변수 목록입니다. **all** 을 사용하면 변수가 모든 호스트에 적용됩니다.

특정 호스트에 변수를 적용하려면 **[hostname:vars]** 을 사용합니다. 예를 들면 **[automationhub:vars]** 입니다.

1.6.4. 인벤토리 파일에서 변수를 선언하는 규칙

문자열 변수 값은 따옴표로 선언됩니다. 예를 들면 다음과 같습니다.

```
pg_database='awx'
pg_username='awx'
pg_password='<password>'
```

:vars 섹션에 선언되면 INI 값이 문자열로 해석됩니다. 예를 들어 **var=FALSE** 는ECDHE과 동일한 문자열을 **만듭니다**. 호스트 행과 달리 **:vars** 섹션은 행당 단일 항목만 허용하므로 **=** 뒤에 있는 모든 항목은 항목의 값이어야 합니다. 호스트 행은 행당 여러 **key=value** 매개변수를 허용합니다. 따라서 공백이 구분자가 아닌 값의 일부임을 나타내는 방법이 필요합니다. 공백이 포함된 값을 따옴표로 묶을 수 있습니다(단일 또는 double). 자세한 내용은 Python shlex 구문 분석 규칙을 참조하십시오.

INI 인벤토리에 설정된 변수 값이 특정 유형(예: 문자열 또는 부울 값)이어야 하는 경우 작업에 필터가 있는 유형을 항상 지정합니다. 변수를 사용할 때 INI 인벤토리에서 설정한 유형에 의존하지 마십시오.

참고

실제 유형의 변수를 혼동하지 않도록 인벤토리 소스에 YAML 형식을 사용하는 것이 좋습니다. YAML 인벤토리 플러그인은 변수 값을 일관되게 그리고 올바르게 처리합니다.

Ansible 인벤토리 파일의 매개변수 값에 #, { 또는 }와 같은 특수 문자가 포함된 경우 값을 이중 이스케이프해야 합니다(단일 및 이중 따옴표로 값 묶임).

예를 들어 **mypasswordwith#hashsigns** 를 변수 **pg_password** 의 값으로 사용하려면 Ansible 호스트 인벤토리 파일에서 **pg_password='"mypasswordwith#hashsigns"** 로 선언합니다.

1.6.5. 인벤토리 파일에서 보안 보안

Ansible Vault를 사용하여 민감한 변수 또는 시크릿 변수를 암호화할 수 있습니다. 그러나 변수 이름과 변수 값을 암호화하면 값의 소스를 찾기가 어렵습니다. 이를 우회하려면 **ansible-vault encrypt_string** 을 사용하여 변수를 개별적으로 암호화하거나 변수가 포함된 파일을 암호화할 수 있습니다.

절차

1. 암호화된 자격 증명을 저장할 **credentials.yml** 레이블이 지정된 파일을 만듭니다.

```
$ cat credentials.yml

admin_password: my_long_admin_pw
pg_password: my_long_pg_pw
registry_password: my_long_registry_pw
```

2. **ansible-vault** 를 사용하여 **credentials.yml** 파일을 암호화합니다.

```
$ ansible-vault encrypt credentials.yml
New Vault password:
Confirm New Vault password:
Encryption successful
```

중요

암호화된 자격 증명 모음 암호를 안전한 위치에 저장합니다.

3. **credentials.yml** 파일이 암호화되었는지 확인합니다.

```
$ cat credentials.yml
$ANSIBLE_VAULT;1.1;
AES256363836965356238653431633333396138333630646533646561383135343531353037646461653937653930633030653234666633306462323630653166663 10a37306230313337633963383130303334313534383962613632303761636632623932653062343839613639653635643365616233313365363661663931386430 0a353239373433313339613465326339313035633565353464356538653631633464343835346432376638623533613666326136343332313163343639393964613265616433363430633534303935646264633034383966336232303365383763
```

4. Ansible Automation Platform 2.2를 설치하려면 **setup.sh** 를 실행하고 **credentials.yml** 과 **--ask-vault-pass 옵션을** 모두 전달합니다.

```
$ ANSIBLE_BECOME_METHOD='sudo' ANSIBLE_BECOME=True
ANSIBLE_HOST_KEY_CHECKING=False ./setup.sh -e @credentials.yml -- --ask-vault-pass
```

1.6.6. 추가 인벤토리 파일 변수

인벤토리 파일에 추가 변수를 포함하여 Red Hat Ansible Automation Platform 설치를 추가로 구성할 수 있습니다. 이러한 구성은 Red Hat Ansible Automation Platform 관리를 위한 선택적 기능을 추가합니다. 텍스트 편집기를 사용하여 인벤토리 파일을 편집하여 이러한 변수를 추가합니다.

인벤토리 파일 변수에 대한 사전 정의된 값의 표는 부록 A: 인벤토리 파일 변수에서 확인할 수 있습니다.

1.7. 지원되는 설치 시나리오

Red Hat은 Red Hat Ansible Automation Platform에 대해 다음과 같은 설치 시나리오를 지원합니다.

1.7.1. 동일한 노드에 데이터베이스가 있는 독립형 자동화 컨트롤러 또는 설치 프로그램이 아닌 데이터베이스

이 시나리오에는 단일 시스템에 웹 프런트 엔드, REST API 백엔드 및 데이터베이스를 포함한 자동화 컨트롤러 설치가 포함됩니다. PostgreSQL을 설치하고 이를 데이터베이스로 사용하도록 자동화 컨트롤러를 구성합니다. 이는 표준 자동화 컨트롤러 설치 시나리오로 간주됩니다.

단일 머신에 Red Hat Ansible Automation Platform 구성 요소 설치의 동일한 노드에 데이터베이스를 사용하여 자동화 컨트롤러 설치를 참조하십시오.

1.7.2. 외부 관리형 데이터베이스가 있는 독립 실행형 자동화 컨트롤러

이 시나리오에는 단일 시스템에 자동화 컨트롤러 서버를 설치하고 원격 PostgreSQL 인스턴스와의 데이터베이스 통신을 구성합니다. 이 원격 PostgreSQL은 관리하는 서버이거나 Amazon RDS와 같은 클라우드 서비스에서 제공할 수 있습니다.

시작하기 위해 *단일 머신에 Red Hat Ansible Automation Platform 구성 요소 설치에서 외부 관리 데이터베이스를 사용하여 자동화 컨트롤러* 설치를 참조하십시오.

1.7.3. 동일한 노드에 있는 데이터베이스가 있는 독립형 자동화 허브 또는 설치 프로그램이 아닌 데이터베이스

이 시나리오에는 단일 시스템에 웹 프런트 엔드, REST API 백엔드 및 데이터베이스를 포함한 자동화 허브 설치가 포함됩니다. PostgreSQL을 설치하고 이를 데이터베이스로 사용하도록 자동화 허브를 구성합니다.

단일 머신에 Red Hat Ansible Automation Platform 구성 요소 설치에서 동일한 노드에 데이터베이스를 사용하여 자동화 허브 설치를 참조하십시오.

1.7.4. 외부 관리 데이터베이스가 있는 독립 실행형 자동화 허브

이 시나리오에는 단일 시스템에 자동화 허브 서버를 설치하고 Red Hat Ansible Automation Platform 설치 프로그램에서 관리하는 원격 PostgreSQL 데이터베이스를 설치합니다.

시작하기 위해 *단일 머신에 Red Hat Ansible Automation Platform 구성 요소 설치에서 외부 데이터베이스를 사용한 자동화 허브* 설치를 참조하십시오.

1.7.5. 자동화 컨트롤러 노드 또는 비installer 관리형 데이터베이스에 데이터베이스를 사용한 플랫폼 설치

이 시나리오에는 자동화 컨트롤러 노드 또는 비installer 관리형 데이터베이스의 데이터베이스와 함께 자동화 컨트롤러 및 자동화 허브를 설치하는 작업이 포함됩니다.

시작하기 위해 Red Hat Ansible Automation Platform 설치에서 자동화 컨트롤러 노드 또는 비installer 관리형 데이터베이스에 데이터베이스를 사용하여 *Red Hat Ansible Automation Platform* 설치를 참조하십시오.

1.7.6. 외부 관리 데이터베이스를 통한 플랫폼 설치

이 시나리오에는 자동화 컨트롤러 및 자동화 허브를 설치하고 원격 PostgreSQL 인스턴스와의 통신을 데이터베이스로 구성합니다. 이 원격 PostgreSQL은 관리하는 서버이거나 Amazon RDS와 같은 클라우드 서비스에서 제공할 수 있습니다.

시작하기 위해 Red Hat Ansible Automation Platform *설치에서 외부 관리 데이터베이스를 사용하여 Red Hat Ansible Automation Platform* 설치를 참조하십시오.

1.7.7. 외부 관리 데이터베이스를 사용한 다중 시스템 클러스터 설치

이 시나리오에는 여러 자동화 컨트롤러 노드 및 자동화 허브 인스턴스를 설치하고 원격 PostgreSQL 인스턴스와의 통신이 데이터베이스로 구성됩니다. 이 원격 PostgreSQL은 관리하는 서버이거나 Amazon RDS와 같은 클라우드 서비스에서 제공할 수 있습니다. 이 시나리오에서는 모든 자동화 컨트롤러가 활성 상태이며 작업을 실행할 수 있으며 모든 노드에서 HTTP 요청을 수신할 수 있습니다.

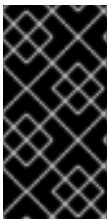
참고

- 클러스터 설정에서 실행하려면 자동화 컨트롤러에서 사용하는 데이터베이스가 external-PostgreSQL인 경우 주 또는 보조 period 노드 중 하나가 아닌 머신에 설치해야 합니다. 중복 설정에서 원격 PostgreSQL 버전 요구 사항은 **PostgreSQL 13** 입니다.
    - 클러스터형 설정 구성에 대한 자세한 내용은 DestinationRule을 참조하십시오. https://docs.ansible.com/automation-controller/latest/html/administration/clustering.html
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소를 제공하여 사용자가 다른 노드의 Private Automation Hub에서 콘텐츠를 동기화할 수 있도록 합니다.

*Multi -machine 클러스터 설치에서 외부 관리형 데이터베이스를 사용하여 다중 노드 Red Hat Ansible Automation Platform 설치*를 참조하십시오.

2장. RED HAT ANSIBLE AUTOMATION PLATFORM 설치

Red Hat Ansible Automation Platform 설치에는 자동화 컨트롤러 및 자동화 허브 배포가 포함됩니다.



중요

Ansible Automation Platform 설치 프로그램을 사용하면 인벤토리당 **하나의 자동화 허브만** 배포할 수 있습니다. Ansible Automation Platform 설치 프로그램을 자동화 허브의 독립 실행형 인스턴스에 사용하고 다양한 인벤토리와 함께 설치 프로그램을 여러 번 실행하여 여러 자동화 허브 노드를 배포할 수 있습니다.

중요

설치 프로그램은 **./setup.sh** 를 실행하기 위해 root로 로그인할 필요가 없습니다. 사용자가 기본 권한 에스컬레이션 방법인 root에 대해 **ANSIBLE_BECOME_METHOD** 환경 변수를 올바르게 구성해야 합니다. 기본 방법은 **sudo** 입니다.

이 설치 옵션에는 다음과 같은 두 가지 시나리오가 포함되어 있습니다.

2.1. 자동화 컨트롤러 노드 또는 비INSTALLER 관리형 데이터베이스에 데이터베이스를 사용하여 RED HAT ANSIBLE AUTOMATION PLATFORM 설치

이러한 지침을 사용하여 자동화 컨트롤러 노드에 있는 데이터베이스 또는 설치 프로그램이 아닌 데이터베이스로 Red Hat Ansible Automation Platform(자동 컨트롤러 및 자동화 허브 모두)을 설치할 수 있습니다.

2.1.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.
- Ansible Automation Platform을 설치하려면 컨테이너 레지스트리 서비스가 필요합니다. 컨테이너 레지스트리에 대한 액세스를 통해 자동화 실행 환경을 Ansible Automation Platform에 로드할 수 있으므로 Ansible 플레이북 및 역할을 실행하기 위한 일관되고 컨테이너화된 환경을 제공할 수 있습니다. 기본적으로 Ansible Automation Platform은 Red Hat 레지스트리 서비스 계정이 필요한 **registry.redhat.io** 를 사용합니다. 레지스트리 서비스 계정을 생성하려면 레지스트리 서비스 계정 생성 가이드를 참조하십시오.

2.1.2. 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. [bundled 설치 프로그램]

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

b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.

3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

2.1.3. 자동화 컨트롤러 노드 또는 비installer 관리형 데이터베이스의 데이터베이스 인벤토리 파일의 예

이 예에서는 Red Hat Ansible Automation Platform을 설치하기 위해 인벤토리 파일을 채우는 방법을 설명합니다. 이 설치 인벤토리 파일에는 자동화 컨트롤러 노드 또는 비installer 관리 데이터베이스의 데이터베이스와 자동화 컨트롤러 및 자동화 허브가 모두 포함되어 있습니다.

중요

- 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치할 수 없습니다.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소를 제공하여 사용자가 다른 노드의 Private Automation Hub에서 콘텐츠를 동기화할 수 있도록 합니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org

[all:vars]
admin_password='<password>'
pg_host=''
pg_port=''
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>'

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='controller.acme.org'
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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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
```

2.1.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 2.1. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.

참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 2.2. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**

Variable	설명	Default
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.1.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

설치가 시작됩니다.

2.1.6. 자동화 컨트롤러 설치 확인

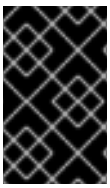
설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

참고

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



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

2.1.6.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 2.3. 자동화 컨트롤러를 구성하는 리소스

link	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러 설정 및 첫 번째 플레이북 실행
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 컨트롤러 설정
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보 관리
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능에 대해 자세히 검토

2.1.7. 자동화 허브 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 허브가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 허브 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 허브에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

2.1.7.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 2.4. 자동화 컨트롤러를 구성하는 리소스

link	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스 구성
자동화 허브에서 Red Hat Certified and Ansible Galaxy 컬렉션 관리	자동화 허브에 콘텐츠 추가
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션 게시

2.1.8. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

2.1.8.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

2.1.8.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

2.1.8.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

2.1.8.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

2.1.8.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배

포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드**를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

2.2. 외부 관리 데이터베이스를 사용하여 RED HAT ANSIBLE AUTOMATION PLATFORM 설치

이러한 지침을 사용하여 외부 관리 데이터베이스와 함께 Red Hat Ansible Automation Platform(자동 컨트롤러 및 자동화 허브 모두)을 설치할 수 있습니다.

2.2.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

2.2.2. 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. [bundled 설치 프로그램]

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

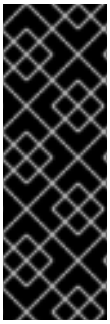
   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

2.2.3. 외부 관리 데이터베이스가 있는 Red Hat Ansible Automation Platform 인벤토리 파일의 예

이 예에서는 Red Hat Ansible Automation Platform을 설치하기 위해 인벤토리 파일을 채우는 방법을 설명합니다. 이 설치 인벤토리 파일에는 외부 관리 데이터베이스가 있는 자동화 컨트롤러와 자동화 허브가 모두 포함되어 있습니다.



중요

- 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치할 수 없습니다.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소를 제공하여 사용자가 다른 노드의 Private Automation Hub에서 콘텐츠를 동기화할 수 있도록 합니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]
controller.acme.org

[automationhub]
automationhub.acme.org

[database]
database-01.acme.org

[all:vars]
admin_password='<password>'
pg_host='database-01.acme.org'
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>'

# Automation Hub Configuration
#
automationhub_admin_password='<password>'
automationhub_pg_host='database-01.acme.org'
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 nginx for the web UI and API
# web_server_ssl_cert=/path/to/tower.cert
# web_server_ssl_key=/path/to/tower.key
# Certificate and key to install in Automation Hub node
# automationhub_ssl_cert=/path/to/automationhub.cert
# automationhub_ssl_key=/path/to/automationhub.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
```

2.2.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 2.5. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.

참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 2.6. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**

Variable	설명	Default
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

2.2.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

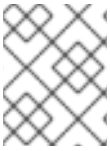
설치가 시작됩니다.

2.2.6. 자동화 컨트롤러 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

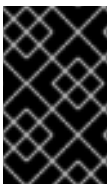
절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.



참고

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



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

2.2.6.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 2.7. 자동화 컨트롤러를 구성하는 리소스

link	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러 설정 및 첫 번째 플레이북 실행
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 컨트롤러 설정
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보 관리
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능에 대해 자세히 검토

2.2.7. 자동화 허브 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 허브가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 허브 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 허브에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

2.2.7.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 2.8. 자동화 컨트롤러를 구성하는 리소스

link	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스 구성
자동화 허브에서 Red Hat Certified and Ansible Galaxy 컬렉션 관리	자동화 허브에 콘텐츠 추가
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션 게시

2.2.8. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

2.2.8.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

2.2.8.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

2.2.8.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

2.2.8.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

2.2.8.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배

포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드**를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

3장. 단일 머신에 RED HAT ANSIBLE AUTOMATION PLATFORM 구성 요소 설치

지원되는 다음 시나리오 중 하나에서 단일 머신에 Red Hat Ansible Automation Platform 구성 요소를 설치할 수 있습니다.

3.1. 동일한 노드에 데이터베이스로 자동화 컨트롤러 설치

이러한 지침을 사용하여 동일한 노드의 데이터베이스 또는 비installer 관리 데이터베이스에 데이터베이스로 자동화 컨트롤러의 독립 실행형 인스턴스를 설치할 수 있습니다. 이 시나리오에는 단일 시스템에 웹 프런트 엔드, REST API 백엔드 및 데이터베이스를 포함한 자동화 컨트롤러 설치가 포함됩니다. PostgreSQL을 설치하고 이를 데이터베이스로 사용하도록 자동화 컨트롤러를 구성합니다. 이는 표준 자동화 컨트롤러 설치 시나리오로 간주됩니다.

3.1.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

3.1.2. 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. [bundled 설치 프로그램]

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

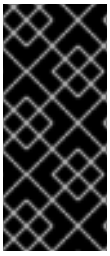
   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

3.1.3. Red Hat Ansible Automation Platform 단일 노드 인벤토리 파일 예

이 예제에서는 자동화 컨트롤러의 단일 노드 설치를 위해 인벤토리 파일을 채우는 방법을 설명합니다.



중요

- **pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 설정이 실패할 수 있습니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]
controller.example.com (1)

[database]

[all:vars]
admin_password='<password>'

pg_host=''
pg_port=''

pg_database='awx'
pg_username='awx'
pg_password='<password>'

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

(1) FQDN/IP로 설정해야 합니다.

3.1.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 3.1. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.

참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 3.2. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음

Variable	설명	Default
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.1.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

설치가 시작됩니다.

3.1.6. 자동화 컨트롤러 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

참고

자동화 컨트롤러 서버는 포트 80(https://<TOWER_SERVER_NAME>/)에서 액세스할 수 있지만 443을 사용할 수 있도록 포트 443으로 리디렉션됩니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

3.1.6.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 3.3. 자동화 컨트롤러를 구성하는 리소스

link	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러 설정 및 첫 번째 플레이북 실행
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 컨트롤러 설정
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보 관리
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능에 대해 자세히 검토

3.1.7. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

3.1.7.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

3.1.7.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

3.1.7.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

3.1.7.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

3.1.7.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드**를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

3.2. 외부 관리 데이터베이스를 사용하여 자동화 컨트롤러 설치

이러한 지침을 사용하여 원격 PostgreSQL 인스턴스와 해당 데이터베이스로 통신하도록 구성된 단일 시스템에 독립 실행형 자동화 컨트롤러 서버를 설치할 수 있습니다. 이 원격 PostgreSQL은 관리하는 서버이거나 Amazon RDS와 같은 클라우드 서비스에서 제공할 수 있습니다.

3.2.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

3.2.2. 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. [bundled 설치 프로그램]

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

   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

3.2.3. 외부 관리 데이터베이스가 있는 독립 실행형 자동화 컨트롤러의 인벤토리 파일 예

이 예제에서는 인벤토리 파일을 채워 자동화 컨트롤러 설치를 외부 데이터베이스로 배포하는 방법을 설명합니다.

중요

- **pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 설정이 실패할 수 있습니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]
controller.example.com (1)


[database]
database.example.com

[all:vars]
admin_password='<password>'
pg_password='<password>'

pg_host='database.example.com'
pg_port='5432'

pg_database='awx'
pg_username='awx'

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

(1) FQDN/IP로 설정해야 합니다.

3.2.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 3.4. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.

참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 3.5. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**

Variable	설명	Default
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.2.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

설치가 시작됩니다.

3.2.6. 자동화 컨트롤러 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

참고

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

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

3.2.6.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 3.6. 자동화 컨트롤러를 구성하는 리소스

link	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러 설정 및 첫 번째 플레이북 실행
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 컨트롤러 설정

link	설명
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보 관리
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능에 대해 자세히 검토

3.2.7. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

3.2.7.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

3.2.7.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

3.2.7.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

3.2.7.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

3.2.7.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드**를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

3.3. 동일한 노드에 데이터베이스로 자동화 허브 설치

이러한 지침을 사용하여 동일한 노드의 데이터베이스 또는 비installer 관리 데이터베이스에 있는 자동화 허브의 독립 실행형 인스턴스를 설치할 수 있습니다.

3.3.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

3.3.2. 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. [bundled 설치 프로그램]

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

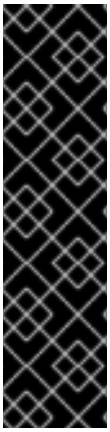
   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

3.3.3. 독립형 자동화 허브 인벤토리 파일 예

이 예제에서는 인벤토리 파일을 채워 자동화 허브의 독립 실행형 인스턴스를 배포하는 방법을 설명합니다.



중요

- Red Hat Ansible Automation Platform 또는 자동화 허브의 경우: **[automationhub]** 그룹에 자동화 허브 호스트 추가. 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치할 수 없습니다.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소 또는 정규화된 도메인 이름(FDQN)을 제공하여 사용자가 다른 노드의 자동화 허브에서 콘텐츠를 동기화하고 설치할 수 있도록 합니다. 'localhost'를 사용하지 마십시오.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]


[automationhub]
127.0.0.1 ansible_connection=local

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host=''
automationhub_pg_port=''

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
```

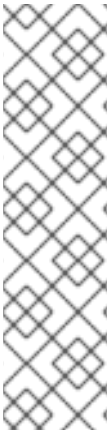
3.3.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 3.7. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.



참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 3.8. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**

Variable	설명	Default
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.3.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

설치가 시작됩니다.

3.3.6. 자동화 허브 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 허브가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 허브 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 허브에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

3.3.6.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 3.9. 자동화 컨트롤러를 구성하는 리소스

link	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스 구성
자동화 허브에서 Red Hat Certified and Ansible Galaxy 컬렉션 관리	자동화 허브에 콘텐츠 추가
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션 게시

3.3.7. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

3.3.7.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

3.3.7.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

3.3.7.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

3.3.7.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

3.3.7.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드** 를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

3.4. 외부 데이터베이스를 사용하여 자동화 허브 설치

이러한 지침을 사용하여 외부 관리 데이터베이스로 자동화 허브의 독립 실행형 인스턴스를 설치할 수 있습니다. 그러면 단일 시스템에 자동화 허브 서버가 설치되고 Ansible Automation Platform 설치 프로그램을 사용하여 원격 PostgreSQL 데이터베이스가 설치됩니다.

3.4.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

3.4.2. 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. [bundled 설치 프로그램]

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

   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

3.4.3. 독립형 자동화 허브 인벤토리 파일 예

이 예제에서는 인벤토리 파일을 채워 자동화 허브의 독립 실행형 인스턴스를 배포하는 방법을 설명합니다.

중요

- Red Hat Ansible Automation Platform 또는 자동화 허브의 경우: '[automationhub] 그룹에 자동화 허브 호스트 추가. 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치할 수 없습니다.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소 또는 정규화된 도메인 이름(FDQN)을 제공하여 사용자가 다른 노드의 자동화 허브에서 콘텐츠를 동기화하고 설치할 수 있도록 합니다. 'localhost'를 사용하지 마십시오.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]


[automationhub]
127.0.0.1 ansible_connection=local

[database]
```

```
host2

[all:vars]
registry_url='registry.redhat.io'
registry_username='<registry username>'
registry_password='<registry password>'

automationhub_admin_password= <PASSWORD>

automationhub_pg_host=''
automationhub_pg_port=''

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
```

3.4.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 3.10. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행

인수	설명
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.

참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 3.11. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**

Variable	설명	Default
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

3.4.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

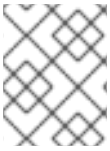
설치가 시작됩니다.

3.4.6. 자동화 컨트롤러 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

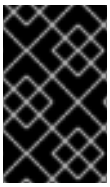
절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.



참고

자동화 컨트롤러 서버는 포트 80(https://<TOWER_SERVER_NAME>/)에서 액세스할 수 있지만 443을 사용할 수 있도록 포트 443으로 리디렉션됩니다.



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

3.4.6.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 3.12. 자동화 컨트롤러를 구성하는 리소스

link	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스 구성
자동화 허브에서 Red Hat Certified and Ansible Galaxy 컬렉션 관리	자동화 허브에 콘텐츠 추가
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션 게시

3.4.7. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

3.4.7.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

3.4.7.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

3.4.7.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

3.4.7.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

3.4.7.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드** 를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

4장. 다중 시스템 클러스터 설치

외부 관리 데이터베이스를 사용하여 자동화 허브를 사용하여 Ansible Automation Platform을 클러스터형 자동화 컨트롤러로 설치할 수 있습니다. 이 모드에서는 여러 자동화 컨트롤러 노드가 설치되고 활성화됩니다. 모든 노드에서 HTTP 요청을 수신할 수 있으며 모든 노드에서 작업을 실행할 수 있습니다. 이렇게 하면 클러스터에 Ansible Automation Platform 서버가 설치되고 PostgreSQL의 원격 인스턴스와 데이터베이스로 통신하도록 구성됩니다. 이 원격 PostgreSQL은 관리하는 서버이거나 Amazon RDS와 같은 클라우드 서비스에서 제공할 수 있습니다.

중요

Ansible Automation Platform 설치 프로그램을 사용하면 인벤토리당 **하나의 자동화 허브만** 배포할 수 있습니다. Ansible Automation Platform 설치 프로그램을 자동화 허브의 독립 실행형 인스턴스에 사용하고 다양한 인벤토리와 함께 설치 프로그램을 여러 번 실행하여 여러 자동화 허브 노드를 배포할 수 있습니다.

4.1. 외부 관리 데이터베이스를 사용하여 다중 노드 RED HAT ANSIBLE AUTOMATION PLATFORM 설치

이러한 지침을 사용하여 Red Hat Ansible Automation Platform을 여러 자동화 컨트롤러 노드로 설치하고 외부 관리 데이터베이스를 통해 자동화 허브를 설치할 수 있습니다.

4.1.1. 사전 요구 사항

- Red Hat Ansible Automation Platform 제품 소프트웨어에서 플랫폼 설치 프로그램을 선택하고 가져옵니다.
- 기본 시스템 요구 사항을 충족하는 머신에 설치 중입니다.
- 레지스트리 서비스 계정 생성 가이드의 지침에 따라 Red Hat 레지스트리 서비스 계정을 생성했습니다.

4.1.2. 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. [bundled 설치 프로그램]

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

   b. [online installer]

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

2. 텍스트 편집기를 사용하여 **인벤토리** 파일을 엽니다.
3. **인벤토리** 파일 매개변수를 편집하여 설치 시나리오를 지정합니다. 아래 예제를 따르십시오.

4.1.3. Red Hat Ansible Automation Platform 다중 노드 인벤토리 파일 예

이 예제에서는 자동화 컨트롤러의 다중 노드 클러스터 설치에 대한 인벤토리 파일을 채우는 방법을 설명합니다.

중요

- 동일한 노드에 자동화 컨트롤러 및 자동화 허브를 설치할 수 없습니다.
- **[automationhub]** 호스트에 연결할 수 있는 IP 주소를 제공하여 사용자가 다른 노드의 Private Automation Hub에서 콘텐츠를 동기화할 수 있도록 합니다.
- **pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 설정이 실패할 수 있습니다.
- **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

```
[automationcontroller]
host1
host11
host12


[automationhub]
host2

[database]
1

[all:vars]
ansible_become=true

admin_password='<password>'

pg_host='dbnode.example.com'
pg_port='5432'

pg_database='tower'
pg_username='tower'
pg_password='<password>'

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

1 필드는 비어 있어야 합니다.

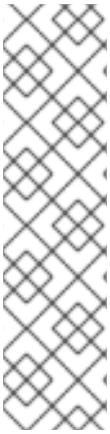
4.1.4. 스크립트 플래그 및 추가 변수 설정

설정 스크립트를 실행할 때 플래그 및 추가 변수를 전달하여 자동화 컨트롤러를 설치할 수도 있습니다.

표 4.1. 플래그

인수	설명
-h	이 도움말 메시지 표시 및 종료
-i INVENTORY_FILE	Ansible 인벤토리 파일의 경로(기본값: **인벤토리**)
-e EXTRA_VARS	추가 Ansible 변수를 key=value 또는 YAML/JSON으로 설정합니다.
-b	설치 대신 데이터베이스 백업을 수행합니다.
-r	설치 대신 데이터베이스 복원 수행
-k	ECDHERET_KEY를 **생성하고** dsitribute

적용할 Ansible 인수를 추가하려면 **--** separator를 사용합니다. 예: **./setup.sh -i my_awesome_inventory.yml -e completionburt_is_country_gold=Trueplan-K**.



참고

- **-r** 을 전달하여 데이터베이스 복원 기본 복원 경로를 수행할 때 EXTRA_VARS가 기본이 아닌 경로가 제공되지 않는 한 기본 복원 기본 복원 경로를 사용합니다. 복원 경로를 지정하는 EXTRA_VAR을 전달하는 아래 예제를 참조하십시오.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

- **-e bundle_install=false** 를 전달하여 온라인 설치를 강제 수행할 수 있습니다.

```
$ ./setup.sh -e bundle_install=false
```

표 4.2. 추가 변수

Variable	설명	Default
upgrade_ansible_with_tower	자동화 컨트롤러를 설치할 때 Ansible도 최신 상태인지 확인합니다.	**False**
create_preload_data	Tower를 설치할 때 Org, 프로젝트, 자격 증명, 작업 템플릿 등도 만듭니다.	**True**
bundle_install_folder	번들에서 설치할 때 번들 리포지토리를 배치해야 합니다.	**var/lib/tower-bundle**

Variable	설명	Default
nginx_disable_https	nginx를 통해 HTTPS 트래픽을 비활성화합니다. 이 기능은 HTTPS를 로드 밸런서로 오프로드하는 경우 유용합니다.	**False**
nginx_disable_hsts	HSTS 웹 보안 정책 메커니즘 비활성화	**False**
nginx_http_port	HTTP를 수신 대기하도록 nginx를 구성하는 포트	**80**
nginx_https_port	HTTPS를 수신 대기하도록 nginx를 구성하는 포트	**443**
backup_dir	백업 시 사용할 임시 위치	**/var/backups/tower/**
restore_backup_file	복원할 대체 백업 파일을 지정합니다.	없음
required_ram	Tower 설치에 필요한 최소 RAM(테스트 설치를 위해서만 변경 가능)	**3750**
min_open_fds	최소 열린 파일 설명 (테스트 설치를 위해서만 변경 가능)	없음
ignore_preflight_errors	템플릿 또는 기타 비 시스템 이미지 (overrides **required_ram** 및 **min_open_fds**)에 설치할 때 유용한 사전 점검을 무시하십시오.	**False**

예

- 코어 업그레이드하려면 다음을 수행합니다.

```
./setup.sh -e upgrade_ansible_with_tower=1
```

- nginx에서 https 처리를 비활성화하려면 다음을 수행합니다.

```
./setup.sh -e nginx_disable_https=true
```

- 백업 파일에서 복원할 때 기본이 아닌 경로를 지정하려면 다음을 수행합니다.

```
./setup.sh -e 'restore_backup_file=/path/to/nondefault/location' -r
```

4.1.5. Red Hat Ansible Automation Platform 설치 프로그램 설정 스크립트 실행

Private Automation Hub를 설치하는 데 필요한 매개변수로 **인벤토리** 파일 업데이트를 완료한 후 설정 스크립트를 실행할 수 있습니다.

절차

1. **setup.sh** 스크립트 실행

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

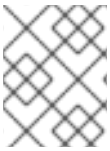
설치가 시작됩니다.

4.1.6. 자동화 컨트롤러 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 컨트롤러가 설치되었는지 확인할 수 있습니다.

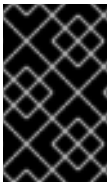
절차

1. **인벤토리** 파일에서 자동화 컨트롤러 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.



참고

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



중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 컨트롤러에 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

4.1.6.1. 추가 자동화 컨트롤러 구성 및 리소스

추가 자동화 컨트롤러 구성을 보려면 다음 리소스를 참조하십시오.

표 4.3. 자동화 컨트롤러를 구성하는 리소스

link	설명
자동화 컨트롤러 빠른 설정 가이드	자동화 컨트롤러 설정 및 첫 번째 플레이북 실행
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 컨트롤러 설정
자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리	Red Hat과 공유하는 자동화 컨트롤러 정보 관리

link	설명
자동화 컨트롤러 사용자 가이드	자동화 컨트롤러 기능에 대해 자세히 검토

4.1.7. 자동화 허브 설치 확인

설치가 완료되면 **인벤토리** 파일에 삽입한 admin 자격 증명으로 로그인하여 자동화 허브가 설치되었는지 확인할 수 있습니다.

절차

1. **인벤토리** 파일에서 자동화 허브 노드에 지정된 IP 주소로 이동합니다.
2. **인벤토리** 파일에서 설정한 관리자 자격 증명으로 로그인합니다.

중요

설치에 실패하고 Red Hat Ansible Automation Platform에 대한 유효한 라이센스를 구매한 고객인 경우 Red Hat 고객 포털 https://access.redhat.com/ 을 통해 Ansible에 문의하십시오.

자동화 허브에 성공적으로 로그인하면 Red Hat Ansible Automation Platform 2.2 설치가 완료되었습니다.

4.1.7.1. 추가 자동화 허브 구성 및 리소스

추가 자동화 허브 구성을 보려면 다음 리소스를 참조하십시오.

표 4.4. 자동화 컨트롤러를 구성하는 리소스

link	설명
프라이빗 자동화 허브에서 사용자 액세스 관리	자동화 허브에 대한 사용자 액세스 구성
자동화 허브에서 Red Hat Certified and Ansible Galaxy 컬렉션 관리	자동화 허브에 콘텐츠 추가
자동화 허브에 독점 콘텐츠 컬렉션 게시	자동화 허브에 내부적으로 개발된 컬렉션 게시

4.1.8. Ansible Automation Platform 2.2의 다음 기능

자동화를 시작하려는 새로운 Ansible Automation Platform 사용자인지 아니면 이전 Ansible 콘텐츠를 최신 설치된 Red Hat Ansible Automation Platform 버전으로 마이그레이션하려는 기존 관리자이든 다음 단계에 따라 Ansible Automation Platform 2.2의 새로운 기능을 활용할 수 있습니다.

4.1.8.1. Ansible Automation Platform 2.2로 데이터 마이그레이션

Ansible Automation Platform 2.2로 업그레이드를 완료하려는 플랫폼 관리자의 경우 새 인스턴스로 데이터를 마이그레이션하는 데 필요한 추가 단계가 있을 수 있습니다.

4.1.8.1.1. 기존 가상 환경(venvs)에서 자동화 실행 환경으로 마이그레이션

Ansible Automation Platform 2.2는 Ansible 자동화를 실행하고 확장하는 데 필요한 구성 요소를 패키지하는 컨테이너화된 이미지인 자동화 실행 환경인 사용자 지정 Python 가상 환경(venvs)에서 벗어나게 합니다. 여기에는 Ansible Core, Ansible Content Collections, Python 종속 항목, Red Hat Enterprise Linux UBI 8 및 추가 패키지 종속 항목이 포함됩니다.

venvs를 실행 환경으로 마이그레이션하려는 경우 (1) **awx-manage** 명령을 사용하여 원래 인스턴스에서 venv 목록을 나열하고 내보낸 다음, (2) **ansible-builder** 를 사용하여 실행 환경을 생성해야 합니다. 자세한 내용은 Automation Execution Environments 가이드 및 Ansible Builder Guide 를 참조하십시오.

4.1.8.1.2. Ansible Builder를 사용하여 Ansible Engine 2.9 이미지로 마이그레이션

Ansible Automation Platform 2.2에서 사용할 Ansible Engine 2.9 이미지를 마이그레이션하기 위해 **ansible-builder** 툴은 자동화 실행 환경에서 사용하기 위해 이미지(사용자 정의 플러그인 및 종속 항목 포함) 프로세스를 자동화합니다. Ansible Builder를 사용하여 실행 환경을 빌드하는 방법에 대한 자세한 내용은 Ansible 빌더 가이드를 참조하십시오.

4.1.8.1.3. Ansible Core 2.13으로 마이그레이션

Ansible Core 2.13으로 업그레이드할 때 최신 버전의 Ansible Core에서 지원하려면 플레이북, 플러그인 또는 Ansible 인프라의 기타 부분을 업데이트해야 합니다. Ansible Core 2.13 호환성을 위한 Ansible 콘텐츠를 업데이트하는 방법은 Ansible-core 2.13 Porting Guide 를 참조하십시오.

4.1.8.2. 자동화 메시를 사용하여 자동화 확장

Red Hat Ansible Automation Platform의 자동화 메시 구성 요소는 다중 사이트 배포 전반에 자동화를 배포하는 프로세스를 단순화합니다. 여러 IT 환경이 분리되어 있는 기업의 경우 자동화 메시는 피어 투 피어 메시 통신 네트워크를 사용하여 실행 노드 전체에 자동화를 배포하고 확장할 수 있는 일관되고 안정적인 방법을 제공합니다.

버전 1.x에서 최신 버전의 Ansible Automation Platform으로 업그레이드할 때 기존 분리된 노드에서 자동화 메시에 필요한 실행 노드로 데이터를 마이그레이션해야 합니다. 하이브리드 및 제어 노드의 네트워크를 계획하여 자동화 메시를 구현한 다음 Ansible Automation Platform 설치 프로그램에 있는 인벤토리 파일을 편집하여 각 실행 노드에 메시 관련 값을 할당할 수 있습니다.

분리된 노드에서 실행 노드로 마이그레이션하는 방법에 대한 지침은 **업그레이드 및 마이그레이션 가이드**를 참조하십시오.

자동화 메시 및 환경에 맞게 자동화 메시를 설계하는 다양한 방법에 대한 자세한 내용은 Red Hat Ansible Automation Platform 자동화 메시 가이드를 참조하십시오.

5장. RED HAT ANSIBLE AUTOMATION PLATFORM에 대한 프록시 지원 구성

프록시를 사용하여 트래픽과 통신하도록 Red Hat Ansible Automation Platform을 구성할 수 있습니다. 프록시 서버는 다른 서버의 리소스를 찾는 클라이언트의 요청에 대한 중개자 역할을 합니다. 클라이언트는 프록시 서버에 연결하여 다른 서버에서 일부 서비스 또는 사용 가능한 리소스를 요청하며, 프록시 서버는 복잡성을 단순화하고 제어하는 방법으로 요청을 평가합니다. 다음 섹션에서는 지원되는 프록시 구성과 설정 방법을 설명합니다.

5.1. 프록시 지원 활성화

프록시 서버 지원을 제공하기 위해 자동화 컨트롤러는 자동화 컨트롤러 설정에서 **REMOTE_HOST_HOST_HEADERS** list 변수를 통해 프록시된 요청(예: ALB, NLB, HAProxy, Squid, Nginx 및 smallproxy)을 처리합니다. 기본적으로 **REMOTE_HOST_HEADERS** 는 **["REMOTE_ADDR", "REMOTE_HOST"]** 로 설정됩니다.

프록시 서버 지원을 활성화하려면 자동화 컨트롤러의 설정 페이지에서 **REMOTE_HOST_HEADERS** 필드를 편집합니다.

절차

1. 자동화 컨트롤러에서 **Settings → Miscellaneous System** 으로 이동합니다.
2. **REMOTE_HOST_HEADERS** 필드에 다음 값을 입력합니다.

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

자동화 컨트롤러는 첫 번째 IP 주소가 있을 때까지 **REMOTE_HOST_HEADERS** 의 헤더 목록을 검색하여 원격 호스트의 IP 주소를 결정합니다.

5.2. 알려진 프록시

자동화 컨트롤러가 **REMOTE_HOST_HEADERS = ['HTTP_X_FORWARDED_FOR', 'REMOTE_ADDR', 'REMOTE_HOST']** 로 설정된 경우 **X-Forwarded-For** 의 값은 자동화 컨트롤러 앞에 묶은 프록시/로드 밸런서에서 시작된 것으로 가정합니다. 프록시/로드 밸런서를 사용하지 않고 자동화 컨트롤러에 연결할 수 있거나 프록시에서 헤더를 확인하지 않는 경우 **X-Forwarded-For** 값은 원래 IP 주소를 위조하기 위해 falsified될 수 있습니다. **REMOTE _HOST_HEADERS 설정에서 HTTP_X_FOR** _FOR을 사용하면 문제가 발생합니다.

이를 방지하려면 자동화 컨트롤러의 설정 메뉴에서 **PROXY_IP_ALLOWED_LIST** 필드를 사용하여 허용되는 알려진 프록시 목록을 구성할 수 있습니다. 알려진 프록시 목록에 없는 로드 밸런서 및 호스트는 거부된 요청을 생성합니다.

5.2.1. 알려진 프록시 구성

자동화 컨트롤러에 대해 알려진 프록시 목록을 구성하려면 자동화 컨트롤러의 설정 페이지의 프록시 IP 주소를 **PROXY_IP_ALLOWED_LIST** 필드에 추가합니다.

절차

1. 자동화 컨트롤러에서 **Settings → Miscellaneous System** 으로 이동합니다.
2. **PROXY_IP_ALLOWED_LIST** 필드에 다음과 같이 자동화 컨트롤러에 연결할 수 있는 IP 주소를 입력합니다.

PROXY_IP_ALLOWED_LIST 항목 예

```
[
  "example1.proxy.com:8080",
  "example2.proxy.com:8080"
]
```

중요

- **PROXY_IP_ALLOWED_LIST** 에는 목록의 프록시가 올바르게 제거되고 클라이언트의 실제 소스 IP와 동일한 **X-Forwarded-For** 값을 올바르게 설정해야 합니다. 자동화 컨트롤러는 **PROXY_IP_ALLOWED_LIST** 의 IP 주소와 호스트 이름을 사용하여 **X-Forwarded-For** 필드에 스푸핑되지 않은 값을 제공할 수 있습니다.
- 다음 조건이 **모두** 충족되지 않는 한 **HTTP_X_FORWARDED_FOR** 을 'REMOTE_HOST_HEADERS'의 항목으로 구성하지 마십시오.
  - ssl 종료와 함께 프록시된 환경을 사용하고 있습니다.
  - 프록시는 클라이언트 스푸핑을 방지하기 위해 **X-Forwarded-For** 헤더의 sanitization 또는 검증 기능을 제공합니다.
  - **/etc/tower/conf.d/remote_host_headers.py** 는 신뢰할 수 있는 프록시 또는 로드 밸런서의 원래 IP 주소만 포함하는 **PROXY_IP_ALLOWED_LIST** 를 정의합니다.

5.3. 역방향 프록시 구성

자동화 컨트롤러 설정에서 **HTTP_X_FORWARDED_FOR** 을 REMOTE_HOST_HEADERS 필드에 추가하여 역방향 프록시 서버 구성을 지원할 수 있습니다. **X-Forwarded-For** (XFF) HTTP 헤더 필드는 HTTP 프록시 또는 로드 밸런서를 통해 웹 서버에 연결하는 클라이언트의 원래 IP 주소를 식별합니다.

절차

1. 자동화 컨트롤러에서 **Settings → Miscellaneous System** 으로 이동합니다.
2. **REMOTE_HOST_HEADERS** 필드에 다음 값을 입력합니다.

```
[
  "HTTP_X_FORWARDED_FOR",
  "REMOTE_ADDR",
  "REMOTE_HOST"
]
```

6장. 자동화 컨트롤러 WEBSOCKET 연결 구성

웹 소켓 구성을 nginx 또는 로드 밸런서 구성과 정렬하도록 자동화 컨트롤러를 구성할 수 있습니다.

6.1. 자동화 컨트롤러를 위한 WEBSOCKET 구성

자동화 컨트롤러 노드는 Websockets를 통해 다른 모든 자동화 컨트롤러 노드에 연결됩니다. 이 상호 연결은 다른 모든 자동화 컨트롤러 노드에 모든 Websocket 출력 메시지를 배포하는 데 사용됩니다. 모든 브라우저 클라이언트 Websocket이 자동화 컨트롤러 노드에서 실행 중일 수 있는 모든 작업을 구독할 수 있으므로 이 작업이 필요합니다. WebSocket 클라이언트는 특정 자동화 컨트롤러 노드로 라우팅되지 않습니다. 모든 자동화 컨트롤러 노드는 모든 Websocket 요청을 처리할 수 있으며 각 자동화 컨트롤러 노드는 모든 클라이언트를 대상으로 하는 모든 Websocket 메시지에 대해 알아야 합니다.

자동화 컨트롤러는 데이터베이스의 인스턴스 레코드를 통해 다른 자동화 컨트롤러 노드 검색을 자동으로 처리합니다.

중요

- 노드는 신뢰할 수 있는 프라이빗 서브넷(오픈 인터넷 제외)에서 Websocket 트래픽을 브로드캐스트하는 것입니다. 따라서 대부분의 Ansible 플레이북 stdout으로 구성된 Websocket 트래픽은 자동화 컨트롤러 노드 간에 암호화되지 않은 상태로 전송됩니다.

6.1.1. 기타 자동화 컨트롤러 노드의 자동 검색 구성

자동화 컨트롤러가 데이터베이스의 인스턴스 레코드를 통해 다른 자동화 컨트롤러 노드 검색을 자동으로 처리할 수 있도록 Websocket 연결을 구성할 수 있습니다.

- 포트, 프로토콜 및 Websocket 연결을 설정할 때 인증서를 확인할지 여부에 대한 자동화 컨트롤러 Websocket 정보를 편집합니다.

```
BROADCAST_WEBSOCKET_PROTOCOL = 'http'
BROADCAST_WEBSOCKET_PORT = 80
BROADCAST_WEBSOCKET_VERIFY_CERT = False
```

7장. 자동화 컨트롤러에서 사용성 분석 및 데이터 수집 관리

자동화 컨트롤러 사용자 인터페이스에서 설정을 옵트아웃하거나 변경하여 사용 편의성 분석 및 데이터 수집 방식을 변경할 수 있습니다.

7.1. 사용 편의성 분석 및 데이터 수집

사용 편의성 데이터 수집은 자동화 컨트롤러 사용자가 구체적으로 자동화 컨트롤러와 상호 작용하는 방법을 더 잘 이해하고 향후 릴리스를 개선하고 사용자 환경을 지속적으로 간소화하기 위해 데이터를 수집하기 위해 자동화 컨트롤러에 포함되어 있습니다.

이 데이터 수집을 위해 자동화 컨트롤러 시험 또는 자동화 컨트롤러의 신규 설치를 설치하는 사용자만 선택할 수 있습니다.

추가 리소스

- 자세한 내용은 Red Hat 개인 정보 보호 정책을 참조하십시오.

7.1.1. 자동화 컨트롤러에서 데이터 수집 제어

설정 메뉴의 **사용자 인터페이스** 탭에서 참여 수준을 설정하여 자동화 컨트롤러가 데이터를 수집하는 방법을 제어할 수 있습니다.

절차

1. 자동화 컨트롤러에 로그인합니다.
2. **설정 → 사용자인터페이스에** 대한 Navgate
3. 사용자 분석 추적 상태 드롭다운 목록에서 원하는 데이터 수집 수준을 선택합니다.
   a. **Off**: 데이터 수집을 방지합니다.
   b. **anonymous**: 특정 사용자 데이터없이 데이터 수집을 활성화합니다.
   c. **자세한**: 특정 사용자 데이터를 포함한 데이터 수집을 활성화합니다.
4. **Save** 를 클릭하여 설정을 적용하거나 **취소** 를 클릭하여 변경 사항을 거부합니다.

8장. 지원되는 인벤토리 플러그인 템플릿

업그레이드 시 기존 구성이 이전 버전과 호환되는 인벤토리 출력을 생성하는 새 형식으로 마이그레이션됩니다. 아래 템플릿을 사용하면 인벤토리를 새 스타일 인벤토리 플러그인 출력으로 마이그레이션하는 데 도움이 됩니다.

8.1. AMAZON WEB SERVICES EC2

```
compose:
  ansible_host: public_ip_address
  ec2_account_id: owner_id
  ec2_ami_launch_index: ami_launch_index | string
  ec2_architecture: architecture
  ec2_block_devices: dict(block_device_mappings | map(attribute='device_name') | list |
zip(block_device_mappings | map(attribute='ebs.volume_id') | list))
  ec2_client_token: client_token
  ec2_dns_name: public_dns_name
  ec2_ebs_optimized: ebs_optimized
  ec2_eventsSet: events | default("")
  ec2_group_name: placement.group_name
  ec2_hypervisor: hypervisor
  ec2_id: instance_id
  ec2_image_id: image_id
  ec2_instance_profile: iam_instance_profile | default("")
  ec2_instance_type: instance_type
  ec2_ip_address: public_ip_address
  ec2_kernel: kernel_id | default("")
  ec2_key_name: key_name
  ec2_launch_time: launch_time | regex_replace(" ", "T") | regex_replace("(\+)(\d\d):(\d)(\d)$",
".\g<2>\g<3>Z")
  ec2_monitored: monitoring.state in ['enabled', 'pending']
  ec2_monitoring_state: monitoring.state
  ec2_persistent: persistent | default(false)
  ec2_placement: placement.availability_zone
  ec2_platform: platform | default("")
  ec2_private_dns_name: private_dns_name
  ec2_private_ip_address: private_ip_address
  ec2_public_dns_name: public_dns_name
  ec2_ramdisk: ramdisk_id | default("")
  ec2_reason: state_transition_reason
  ec2_region: placement.region
  ec2_requester_id: requester_id | default("")
  ec2_root_device_name: root_device_name
  ec2_root_device_type: root_device_type
  ec2_security_group_ids: security_groups | map(attribute='group_id') | list |  join(',')
  ec2_security_group_names: security_groups | map(attribute='group_name') | list |  join(',')
  ec2_sourceDestCheck: source_dest_check | default(false) | lower | string
  ec2_spot_instance_request_id: spot_instance_request_id | default("")
  ec2_state: state.name
  ec2_state_code: state.code
  ec2_state_reason: state_reason.message if state_reason is defined else ""
  ec2_subnet_id: subnet_id | default("")
  ec2_tag_Name: tags.Name
  ec2_virtualization_type: virtualization_type
  ec2_vpc_id: vpc_id | default("")
```

```
filters:
  instance-state-name:
  - running
groups:
  ec2: true
hostnames:
  - network-interface.addresses.association.public-ip
  - dns-name
  - private-dns-name
keyed_groups:
  - key: image_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: images
    prefix: ''
    separator: ''
  - key: placement.availability_zone
    parent_group: zones
    prefix: ''
    separator: ''
  - key: ec2_account_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: accounts
    prefix: ''
    separator: ''
  - key: ec2_state | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: instance_states
    prefix: instance_state
  - key: platform | default("undefined") | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: platforms
    prefix: platform
  - key: instance_type | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: types
    prefix: type
  - key: key_name | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: keys
    prefix: key
  - key: placement.region
    parent_group: regions
    prefix: ''
    separator: ''
  - key: security_groups | map(attribute="group_name") | map("regex_replace", "[^A-Za-z0-9\_]", "_") |
list
    parent_group: security_groups
    prefix: security_group
  - key: dict(tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list | zip(tags.values()
      | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list))
    parent_group: tags
    prefix: tag
  - key: tags.keys() | map("regex_replace", "[^A-Za-z0-9\_]", "_") | list
    parent_group: tags
    prefix: tag
  - key: vpc_id | regex_replace("[^A-Za-z0-9\_]", "_")
    parent_group: vpcs
    prefix: vpc_id
  - key: placement.availability_zone
    parent_group: '{{ placement.region }}'
    prefix: ''
```

```
    separator: ''
plugin: amazon.aws.aws_ec2
use_contrib_script_compatible_sanitization: true
```

8.2. GOOGLE COMPUTE ENGINE

```
auth_kind: serviceaccount
compose:
  ansible_ssh_host: networkInterfaces[0].accessConfigs[0].natIP |
default(networkInterfaces[0].networkIP)
  gce_description: description if description else None
  gce_id: id
  gce_image: image
  gce_machine_type: machineType
  gce_metadata: metadata.get("items", []) | items2dict(key_name="key", value_name="value")
  gce_name: name
  gce_network: networkInterfaces[0].network.name
  gce_private_ip: networkInterfaces[0].networkIP
  gce_public_ip: networkInterfaces[0].accessConfigs[0].natIP | default(None)
  gce_status: status
  gce_subnetwork: networkInterfaces[0].subnetwork.name
  gce_tags: tags.get("items", [])
  gce_zone: zone
hostnames:
- name
- public_ip
- private_ip
keyed_groups:
- key: gce_subnetwork
  prefix: network
- key: gce_private_ip
  prefix: ''
  separator: ''
- key: gce_public_ip
  prefix: ''
  separator: ''
- key: machineType
  prefix: ''
  separator: ''
- key: zone
  prefix: ''
  separator: ''
- key: gce_tags
  prefix: tag
- key: status | lower
  prefix: status
- key: image
  prefix: ''
  separator: ''
plugin: google.cloud.gcp_compute
retrieve_image_info: true
use_contrib_script_compatible_sanitization: true
```

8.3. MICROSOFT AZURE RESOURCE MANAGER

```
conditional_groups:
  azure: true
default_host_filters: []
fail_on_template_errors: false
hostvar_expressions:
  computer_name: name
  private_ip: private_ipv4_addresses[0] if private_ipv4_addresses else None
  provisioning_state: provisioning_state | title
  public_ip: public_ipv4_addresses[0] if public_ipv4_addresses else None
  public_ip_id: public_ip_id if public_ip_id is defined else None
  public_ip_name: public_ip_name if public_ip_name is defined else None
  tags: tags if tags else None
  type: resource_type
keyed_groups:
- key: location
  prefix: ''
  separator: ''
- key: tags.keys() | list if tags else []
  prefix: ''
  separator: ''
- key: security_group
  prefix: ''
  separator: ''
- key: resource_group
  prefix: ''
  separator: ''
- key: os_disk.operating_system_type
  prefix: ''
  separator: ''
- key: dict(tags.keys() | map("regex_replace", "^(.*)$", "\1_") | list | zip(tags.values() | list)) if tags else
[]
  prefix: ''
  separator: ''
plain_host_names: true
plugin: azure.azcollection.azure_rm
use_contrib_script_compatible_sanitization: true
```

8.4. VMWARE VCENTER

```
compose:
  ansible_host: guest.ipAddress
  ansible_ssh_host: guest.ipAddress
  ansible_uuid: 99999999 | random | to_uuid
  availablefield: availableField
  configissue: configIssue
  configstatus: configStatus
  customvalue: customValue
  effectiverole: effectiveRole
  guestheartbeatstatus: guestHeartbeatStatus
  layoutex: layoutEx
  overallstatus: overallStatus
  parentvapp: parentVApp
  recenttask: recentTask
  resourcepool: resourcePool
  rootsnapshot: rootSnapshot
```

```
  triggeredalarmstate: triggeredAlarmState
filters:
- runtime.powerState == "poweredOn"
keyed_groups:
- key: config.guestId
  prefix: ''
  separator: ''
- key: '"templates" if config.template else "guests"'
  prefix: ''
  separator: ''
plugin: community.vmware.vmware_vm_inventory
properties:
- availableField
- configIssue
- configStatus
- customValue
- datastore
- effectiveRole
- guestHeartbeatStatus
- layout
- layoutEx
- name
- network
- overallStatus
- parentVApp
- permission
- recentTask
- resourcePool
- rootSnapshot
- snapshot
- triggeredAlarmState
- value
- capability
- config
- guest
- runtime
- storage
- summary
strict: false
with_nested_properties: true
```

8.5. RED HAT SATELLITE 6

```
group_prefix: foreman_
keyed_groups:
- key: foreman['environment_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_') |
regex_replace('none', '')
  prefix: foreman_environment_
  separator: ''
- key: foreman['location_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_location_
  separator: ''
- key: foreman['organization_name'] | lower | regex_replace(' ', '') | regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_organization_
  separator: ''
```

```
- key: foreman['content_facet_attributes']['lifecycle_environment_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_lifecycle_environment_
  separator: ''
- key: foreman['content_facet_attributes']['content_view_name'] | lower | regex_replace(' ', '') |
regex_replace('[^A-Za-z0-9_]', '_')
  prefix: foreman_content_view_
  separator: ''
legacy_hostvars: true
plugin: theforeman.foreman.foreman
validate_certs: false
want_facts: true
want_hostcollections: false
want_params: true
```

8.6. OPENSTACK

```
expand_hostvars: true
fail_on_errors: true
inventory_hostname: uuid
plugin: openstack.cloud.openstack
```

8.7. RED HAT VIRTUALIZATION

```
compose:
  ansible_host: (devices.values() | list)[0][0] if devices else None
keyed_groups:
- key: cluster
  prefix: cluster
  separator: _
- key: status
  prefix: status
  separator: _
- key: tags
  prefix: tag
  separator: _
ovirt_hostname_preference:
- name
- fqdn
ovirt_insecure: false
plugin: ovirt.ovirt.ovirt
```

8.8. 자동화 컨트롤러

```
include_metadata: true
inventory_id: <inventory_id or url_quoted_named_url>
plugin: awx.awx.tower
validate_certs: <true or false>
```

9장. 사용자 정의 알림에 대해 지원되는 속성

이 섹션에서는 지원되는 작업 특성 목록과 알림에 대한 메시지 텍스트를 구성하는 적절한 구문에 대해 설명합니다. 지원되는 작업 특성은 다음과 같습니다.

- **allow_simultaneous** - (boolean) 이 작업과 연결된 JT에서 여러 작업을 동시에 실행할 수 있는지 나타냅니다.
- **controller_node** - 격리된 실행 환경을 관리하는 인스턴스
- **작성 - 이 작업이 생성** 될 때 (datetime) 타임 스탬프
- **custom_virtualenv** - 작업을 실행하는 데 사용되는 사용자 지정 가상 환경
- **description** - (문자열) 작업에 대한 선택적 설명
- **diff_mode** - (boolean) 활성화된 경우 호스트의 템플릿 파일에 대한 텍스트 변경 사항이 표준 출력에 표시됩니다.
- **elapsed** - 작업이 실행된 시간(초) 경과
- **execution_node** - 작업이 실행된 노드
- **failed** - 작업이 실패한 경우 (boolean) true
- **finished** - (datetime) 작업 실행을 완료한 날짜 및 시간
- **force_handlers** - (boolean) 핸들러가 강제 적용되면 해당 호스트에서 작업이 실패하더라도 알림을 받을 수 있습니다(예: 연결할 수 없는 호스트)는 핸들러가 실행되지 않도록 할 수 있습니다.
- **포크** - (int) 작업에 필요한 포크 수
- **ID** - 이 작업의 데이터베이스 ID
- **job_explanation** - (문자열) status 필드를 실행하여 stdout을 실행하고 캡처할 수 없는 경우 작업의 상태를 나타냅니다.
- **job_slice_count** - (integer) 슬라이스 작업의 일부로 실행하는 경우 총 슬라이스 수입니다(작업이 슬라이스된 작업의 일부가 아닌 경우)
- **job_slice_number** - (integer) 슬라이스 작업의 일부로 실행하는 경우, 운영되는 인벤토리 슬라이스의 ID입니다(sliced 작업의 일부가 아닌 경우, 속성이 사용되지 않음)
- **job_tags** - (문자열) 지정된 태그가 있는 작업만 실행합니다.
- **job_type** - (선택 사항) 실행, 확인 또는 검사
- **launch_type** - (선택) 매뉴얼, relaunch, callback, scheduled, dependency, workflow, sync, scm
- **limit** - (문자열) 이 호스트 집합으로 제한되는 경우
- **수정 - 이 작업이 마지막으로 수정된** 시간(datetime) 타임 스탬프
- **name** - (문자열) 이 작업의 이름
- **playbook** - (문자열) 플레이북 실행

- **scm_revision** - (문자열) 이 작업에 사용된 프로젝트의 리비전(사용 가능한 경우)
- **skip_tags** - (문자열) 플레이북 실행이 지정된 경우 이 태그 세트를 건너뜁니다.
- **start_at_task** - (문자열) 플레이북 실행이 지정된 경우 이 이름과 일치하는 작업에서 시작됩니다.
- **Started** - 작업을 시작하기 위해 대기한 날짜 및 시간입니다.
- **status** - (선택) new, pending, waiting, running, successful, failed, error, canceled
- **timeout** - 작업이 취소되기 전에 실행할 시간(초)입니다.
- **유형** - 이 작업에 대한 데이터 유형
- **url** - 이 작업의 URL
- **use_fact_cache** - 작업에 활성화된 경우 Tower는 Ansible Fact Cache 플러그인 역할을 하며, 플레이북 실행 끝에 팩트를 유지하고 Ansible에서 사용할 팩트를 캐싱합니다.
- **상세 정보** - (선택 사항) 0 ~ 5 (WinRM Debug를 통해 Normal에 대응)
- **host_status_counts** (각 상태에 고유하게 할당된 호스트의 개수)
  - **건너뛰기** (integer)
  - **확인** (integer)
  - **변경됨** (integer)
  - **실패** (integer)
  - **어두운** (integer)
  - **처리** (integer)
  - **구조** 됨(integer)
  - **무시됨** (integer)
  - **실패** (boolean)
- **summary_fields:**
  - **인벤토리**
    - **ID** - 인벤토리의 데이터베이스 ID
    - **name** - (문자열) 인벤토리의 이름
    - **description** - (문자열) 인벤토리에 대한 선택적 설명
    - **has_active_failures** - (boolean) (더 이상 사용되지 않음) 이 인벤토리의 호스트가 실패했는지 여부를 나타내는 플래그
    - **total_hosts** - (더 이상 사용되지 않음) 이 인벤토리의 호스트 수입니다.
    - **hosts_with_active_failures** - (더 이상 사용되지 않음) 이 인벤토리의 호스트 수에 활성 오류가 발생했습니다.

- **total_groups** - (더 이상 사용되지 않음) 이 인벤토리의 총 그룹 수
- **groups_with_active_failures** - (더 이상 사용되지 않음) 이 인벤토리의 호스트 수(활성 실패)
- **has_inventory_sources** - (더 이상 사용되지 않음) 이 인벤토리에 외부 인벤토리 소스가 있는지 여부를 나타내는 플래그
- **total_inventory_sources** - (int) 이 인벤토리 내에 구성된 외부 인벤토리 소스의 총 수
- **inventory_sources_with_failures** - (int) 실패와 함께 이 인벤토리의 외부 인벤토리 소스 수
- **organization_id** - (id) 이 인벤토리를 포함하는 조직
- **kind** - (선택 사항) (호스트를 나타내는 경우 인벤토리와의 직접 링크가 있음) 또는 'smart'

project

- **ID** - (int) 프로젝트의 데이터베이스 ID
- **name** - (문자열) 프로젝트의 이름
- **description** - (문자열) 프로젝트에 대한 선택적 설명
- **status** - (선택) 새로운 보류중인, waiting, running, successful, failed, error, canceled, never updated, ok, missing 중 하나입니다.
- **scm_type** (선택 사항) - git, hg, svn, insights 중 하나

job_template

- **ID** - 작업 템플릿의 데이터베이스 ID
- **Name** - (문자열) 작업 템플릿의 이름
- **description** - (문자열) 작업 템플릿에 대한 선택적 설명

unified_job_template

- **ID** - (int) 통합 작업 템플릿의 데이터베이스 ID
- **name** - (문자열) 통합 작업 템플릿의 이름
- **description** - (문자열) 통합 작업 템플릿에 대한 선택적 설명
- **unified_job_type** - (선택) 통합 작업 유형(job, workflow_job, project_update 등)

instance_group

- **ID** - (int) 인스턴스 그룹의 데이터베이스 ID
- **Name** - (문자열) 인스턴스 그룹의 이름

created_by

- 작업을 시작한 사용자의 (int) 데이터베이스 ID

- 작업을 시작한 사용자 이름(문자열)
- first_name - (문자열) 첫 번째 이름
- last_name - (문자열) 마지막 이름

- labels
  - count - (int) 라벨 수
  - results - 라벨을 나타내는 사전 목록(예: {"id": 5, "name": "database jobs"})

작업에 대한 정보는 그룹화된 중괄호 {{ }}를 사용하여 사용자 정의 알림 메시지에서 참조할 수 있습니다. 특정 작업 속성은 점 표기법을 사용하여 액세스할 수 있습니다(예: {{ job.summary_fields.inventory.name }}). 일부 설명자를 나타내기 위해 앞 또는 일반 텍스트를 중심으로 사용하는 모든 문자(예: 작업 ID의 경우 '#' 및 단일-ECDHE)를 위해 설명을 위해 추가할 수 있습니다. 사용자 정의 메시지는 메시지 전체에서 여러 변수를 포함할 수 있습니다.

```
{{ job_friendly_name }} {{ job.id }} ran on {{ job.execution_node }} in {{ job.elapsed }} seconds.
```

작업 특성 외에도 기타 몇 가지 변수를 템플릿에 추가할 수 있습니다.

approval_node_name - (문자열) 승인 노드 이름

approval_status - 승인, 거부, timed_out 중 하나

url - (문자열) 알림이 발송되는 작업의 URL (시작, 성공, 실패, 승인 알림에 적용됩니다)

workflow_url - 관련 승인 노드에 대한 URL입니다. 이를 통해 알림 수신자가 관련 워크플로 작업 페이지로 이동하여 수행할 내용을 확인할 수 있습니다(즉, 이 노드는 {{ workflow_url }}에서 볼 수 있습니다). 승인 관련 알림의 경우 url 및 workflow_url 둘 다 동일합니다.

job_friendly_name - (문자열) 작업의 친숙한 이름

job_metadata - (문자열) 작업 메타데이터는 JSON 문자열로, 예를 들면 다음과 같습니다.

```
{'url': 'https://towerhost/$/jobs/playbook/13',
 'traceback': '',
 'status': 'running',
 'started': '2019-08-07T21:46:38.362630+00:00',
 'project': 'Stub project',
 'playbook': 'ping.yml',
 'name': 'Stub Job Template',
 'limit': '',
 'inventory': 'Stub Inventory',
 'id': 42,
 'hosts': {},
 'friendly_name': 'Job',
 'finished': False,
 'credential': 'Stub credential',
 'created_by': 'admin'}
```

부록 A. 인벤토리 파일 변수

다음 표에는 **Ansible** 설치 인벤토리 파일에서 사용되는 사전 정의된 변수에 대한 정보가 포함되어 있습니다.

이러한 변수가 모두 필요한 것은 아닙니다.

A.1. 일반 변수

변수	설명
enable_insights_collection	노드가 Subscription Manager에 등록된 경우 기본 설치는 Red Hat Insights for Red Hat Ansible Automation Platform Service에 노드를 등록합니다. 비활성화하려면 **False** 로 설정합니다. 기본값 = **true**
registry_password	**registry_url** 에 액세스하기 위한 암호 인증 정보 **[automationcontroller]** 및 **[automationhub]** 그룹에 모두 사용됩니다. **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 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	[자동 **컨트롤러] 및 [자동 허브]** 그룹 모두에 사용됩니다. 기본값 = **registry.redhat.io**.
registry_username	**registry_url** 에 액세스하기 위한 사용자 인증 정보 **[automationcontroller]** 및 **[automationhub]** 그룹에 모두 사용되지만 **registry_url** 값이 **registry.redhat.io** 인 경우에만 사용됩니다. **registry_username** 및 **registry_password** 에 Red Hat Registry Service Account 자격 증명을 입력하여 Red Hat 컨테이너 레지스트리에 연결합니다.

A.2. ANSIBLE 자동화 허브 변수

변수	설명
automationhub_admin_password	필수 항목
automationhub_api_token	Ansible Automation Platform 2.0 또는 이전 버전에서 업그레이드하는 경우 다음 중 하나를 수행해야 합니다. ● 기존 Ansible 자동화 허브 토큰을 **automationhub_api_token** 으로 제공 ● 새 토큰을 생성하려면 **generate_automationhub_token** 을 **true** 로 설정합니다. 새 토큰을 생성하면 기존 토큰이 무효화됩니다.
automationhub_authentication_backend	이 변수는 기본적으로 설정되지 않습니다. LDAP 인증을 사용하려면 **ldap** 로 설정합니다. **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_seach_base_dn**
automationhub_auto_sign_collections	컬렉션 서명 서비스가 활성화된 경우 컬렉션은 기본적으로 자동으로 서명되지 않습니다. 이 매개변수를 **true** 로 설정하면 기본적으로 서명합니다. default = **false**.
automationhub_backup_collections	*선택 사항* Ansible 자동화 허브는 **/var/lib/pulp** 에 아티팩트를 제공합니다. 자동화 컨트롤러는 기본적으로 아티팩트를 자동으로 백업합니다. **automationhub_backup_collections = false** 를 설정할 수도 있고 backup/restore 프로세스는 **/var/lib/pulp** 를 백업하거나 복원하지 않습니다. 기본값 = **true**

변수	설명
automationhub_collection_signing_service_key	컬렉션 서명 서비스가 활성화된 경우 컬렉션에 올바르게 서명될 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/key/to/sign**
automationhub_collection_signing_service_script	컬렉션 서명 서비스가 활성화된 경우 컬렉션에 올바르게 서명될 수 있도록 이 변수를 제공해야 합니다. **/absolute/path/to/script/that/signs**
automationhub_create_default_collection_signing_service	기본 설치에서는 서명 서비스를 생성하지 않습니다. **true** 로 설정하면 서명 서비스가 생성됩니다. 기본값 = **false**
automationhub_disable_hsts	기본 설치는 TLS 지원 Ansible 자동화 허브를 배포합니다. HSTS(*HTTP Strict Transport Security*) 웹 보안 정책이 활성화된 경우 자동화 허브를 사용합니다. 달리 지정하지 않는 한 HSTS 웹 보안 정책 메커니즘이 활성화됩니다. 이 설정을 사용하면 필요한 경우 비활성화할 수 있습니다. 기본값 = **false**
automationhub_disable_https	*선택 사항* Ansible 자동화 허브가 HTTPS를 활성화된 경우 다음을 수행합니다. default = **false**.
automationhub_enable_api_access_log	**true** 로 설정하면 사용자 이름 및 IP 주소를 포함하여 플랫폼에 수행된 모든 사용자 작업을 기록하는 **/var/log/galaxy_api_access.log** 에 로그 파일을 생성합니다. default = **false**.
automationhub_importer_settings	*선택 사항* import-importer로 전달되도록 설정의 사전입니다. 가져오기 시 컬렉션은 일련의 검사를 통과할 수 있습니다. 이 동작은 이름이 sensitive **-importer.cfg 구성에** 의해 구동됩니다. 예를 들면 **ansible-doc**,**ansible-lint**, **flake8** 입니다. 이 매개변수를 사용하면 이 구성을 구동할 수 있습니다.

Ansible 자동화 허브가 LDAP에 직접 연결하려면 다음 변수를 구성해야 합니다. ldap_extra_settings 변수를 사용하여 전달할 수 있는 다른 LDAP 관련 변수(아래의 automationhub_ldap_xxx 변수에서 다루지 않음) 목록은 https://django-auth-ldap.readthedocs.io/en/latest/reference.html#settings에서 확인할 수 있습니다.

변수	설명
automationhub_ldap_bind_dn	**automationhub_ldap_bind_password** 를 사용하여 LDAP 서버에 바인딩할 때 사용할 이름입니다.
automationhub_ldap_bind_password	*필수 항목* **automationhub_ldap_bind_dn** 과 함께 사용할 암호입니다.
automationhub_ldap_group_search_base_dn	사용자가 속할 수 있는 모든 LDAP 그룹을 찾는 LDAPSearch 오브젝트입니다. 구성에서 LDAP 그룹을 참조하는 경우, 이 및 **automationhub_ldap_group_type** 을 설정해야 합니다. 기본값 = **없음**
automatiohub_ldap_group_search_filter	*선택 사항* 필터를 검색하여 그룹 멤버십을 찾습니다. Default = **(objectClass=Group)**
automationhub_ldap_group_search_scope	*선택 사항* 기본값 = **SUBTREE**
automationhub_ldap_group_type_class	*선택 사항* Default =**django_auth_ldap.config:GroupOfNamesType**
automationhub_ldap_server_uri	LDAP 서버의 URI입니다. 기본 LDAP 라이브러리에서 지원하는 모든 URI일 수 있습니다.
automationhub_ldap_user_search_base_dn	디렉터리에서 사용자를 찾는 LDAPSearch 오브젝트입니다. filter 매개변수에는 사용자 이름의 자리 표시자 %(user)가 포함되어야 합니다. 인증이 성공하려면 정확히 하나의 결과를 반환해야 합니다.

변수	설명
automationhub_main_url	Single Sign-On을 사용하는 경우 클라이언트가 연결할 기본 자동화 허브 URL을 지정합니다(예: **https://<hubaddress.example.com** >). 이 URL은 외부 주소가 **/etc/pulp/settings.py** 에 입력됩니다. 지정하지 않으면 **[automationhub]** 그룹의 첫 번째 노드가 사용됩니다.
automationhub_pg_database	*필수 항목* 데이터베이스 이름입니다. default = **automationhub**
automationhub_pg_host	내부 데이터베이스를 사용하지 않는 경우 필수 항목입니다.
automationhub_pg_password	자동화 허브 PostgreSQL 데이터베이스의 암호입니다. **automationhub_pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 암호가 실패할 수 있습니다.
automationhub_pg_port	내부 데이터베이스를 사용하지 않는 경우 필수 항목입니다. 기본값 = 5432
automationhub_pg_sslmode	필수 항목입니다. default = **prefer**
automationhub_pg_username	필수 항목 default = **automationhub**
automationhub_require_content_approval	*선택 사항* 자동화 허브가 컬렉션을 제공하기 전에 승인 메커니즘을 적용합니다. 기본적으로 자동화 허브에 컬렉션을 업로드할 때 관리자는 사용자가 사용할 수 있도록 만들기 전에 승인해야 합니다. 콘텐츠 승인 흐름을 비활성화하려면 변수를 **false** 로 설정합니다. 기본값 = **true**

변수	설명
automationhub_ssl_cert	*선택 사항* **/path/to/automationhub.cert** 는 **web_server_ssl_cert** 와 동일하지만 자동화 허브 UI 및 API용
automationhub_ssl_key	*선택 사항* **/path/to/automationhub.key** **web_server_ssl_key** 와 동일하지만 자동화 허브 UI 및 API
automationhub_ssl_validate_certs	Red Hat Ansible Automation Platform 2.2 이상에서는 이 값이 더 이상 사용되지 않습니다. 자동화 허브가 기본적으로 자체 인증서를 요청할 때 인증서의 유효성을 검사해야 하는 경우 Ansible Automation Platform은 자체 서명된 인증서로 배포됩니다. default = **false**.
generate_automationhub_token	Red Hat Ansible Automation Platform 2.0 또는 이전 버전에서 업그레이드하는 경우 다음 중 하나를 수행해야 합니다. ● 기존 Ansible 자동화 허브 토큰을 **automationhub_api_token** 으로 제공 ● 새 토큰을 생성하려면 **generate_automationhub_token** 을 **true** 로 설정합니다. 새 토큰을 생성하면 기존 토큰이 무효화됩니다.
pulp_db_fields_key	가져오려는 대칭 암호화 키의 상대 경로 또는 절대 경로입니다. 경로는 Ansible 관리 노드에 있습니다. 데이터베이스의 특정 필드(예: 자격 증명)를 암호화하는 데 사용됩니다. 지정하지 않으면 새 키가 생성됩니다.

A.3. RED HAT SINGLE SIGN-ON 변수

*** automationhub 또는 automationcatalog 에 대해 이러한 변수를 사용하십시오.**

변수	설명
***_sso_console_admin_password**	SSO 관리자 암호.

변수	설명
***_sso_console_keystore_file**	SSO 노드에 설치할 키 저장소 파일입니다. **/path/to/sso.jks**
***_sso_host**	자동화 서비스 카탈로그에는 인증을 위해 SSO 및 SSO 관리자 자격 증명이 필요합니다. 애플리케이션에 필요한 자동화 서비스 카탈로그별 역할을 설정하는 데 SSO 관리 자격 증명이 필요합니다. 구성을 위해 SSO가 인벤토리에 제공되지 않는 경우, 이 변수를 사용하여 SSO 호스트를 정의해야 합니다.
***_sso_keystore_password**	https 지원 SSO에는 키 저장소 암호가 필요합니다. 기본 설치는 **sso_use_https=True**를 사용하여 SSO를 배포합니다.
***_sso_use_https**	Single Sign On 사용 중인 경우

A.4. 자동화 서비스 카탈로그 변수

변수	설명
automationcatalog_controller_password	컨트롤러 호스트에서 토큰을 생성하는 데 사용됩니다. **automation_controller_main_url** 도 정의해야 합니다.
automationcatalog_controller_token	자동화 컨트롤러에 대해 미리 생성된 OAuth 토큰에 사용됩니다. 이 토큰은 토큰을 생성하는 대신 사용됩니다.
automationcatalog_controller_username	컨트롤러 호스트에서 토큰을 생성하는 데 사용됩니다. **automation_controller_main_url** 도 정의해야 합니다.
automationcatalog_controller_verify_ssl	자동화 서비스 카탈로그에서 자동화 컨트롤러까지 SSL 검증을 활성화하거나 비활성화하는 데 사용됩니다. default = **true**.
automationcatalog_disable_hsts	자동화 서비스 카탈로그에 대한 HSTS 웹 보안 정책을 활성화 또는 비활성화하는 데 사용됩니다. 기본값 = 'false.

변수	설명
automationcatalog_disable_https	서비스 카탈로그에 대한 HSTS 웹 보안 정책을 활성화 또는 비활성화하는 데 사용됩니다. default = **false**.
automationcatalog_enable_analytics_collection	자동화 서비스 카탈로그에 대한 분석 컬렉션의 활성화 제어
automationcatalog_main_url	SSO와 자동화 서비스 카탈로그 호스트 간에 사용해야 하는 대체 호스트 이름이 있는 경우 Red Hat Single Sign-On 호스트 구성에서 사용합니다.
automationcatalog_pg_database	자동화 서비스 카탈로그의 데이터베이스 URL입니다.
automationcatalog_pg_host	자동화 서비스 카탈로그의 PostgreSQL 호스트(데이터베이스 노드)
automationcatalog_pg_password	자동화 서비스 카탈로그의 PostgreSQL 데이터베이스의 암호입니다. **automationcatalog_pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 암호가 실패할 수 있습니다.
automationcatalog_pg_port	자동화 서비스 카탈로그에 사용할 PostgreSQL 포트입니다. 기본값 = 5432
automationcatalog_pg_username	자동화 서비스 카탈로그의 postgres ID입니다.
automationcatalog_ssl_cert	사용자 정의 제공 SSL 인증서 파일의 경로입니다. **automationcatalog_ssl_key** 가 필요합니다. 내부적으로 관리되는 CA 서명은 제공되지 않고 https가 활성화된 경우 인증서를 생성합니다.
automationcatalog_ssl_key	사용자 정의 제공 SSL 인증서 키 파일의 경로입니다. **automationcatalog_ssl_cert** 가 필요합니다. 내부적으로 관리되는 CA 서명 및 제공되지 않는 경우 인증서를 생성합니다.

A.5. 자동화 컨트롤러 변수

변수	설명
admin_password	설치 완료 시 UI에 액세스할 관리 사용자의 암호입니다.

변수	설명
automationcontroller_main_url	자동화 서비스 카탈로그를 통한 SSO 구성에 필요한 대체 프런트 엔드 URL의 경우 URL을 제공합니다. 자동화 서비스 카탈로그를 사용하려면 자동화 컨트롤러를 사용하여 컨트롤러를 설치하거나 활성 및 라우팅 가능한 컨트롤러 서버에 대한 URL을 이 변수를 제공해야 합니다.
automationcontroller_password	자동화 컨트롤러 인스턴스의 암호입니다.
automationcontroller_username	자동화 컨트롤러 인스턴스의 사용자 이름입니다.
node_state	*선택 사항* 노드 또는 노드 그룹의 상태. 유효한 옵션은 **활성** 입니다. 클러스터 또는 **iso_migrate** 에서 노드를 삭제하여 기존 분리된 노드를 실행 노드로 마이그레이션하는**deprovision** 입니다. 기본값 = **active**.
node_type	**[automationcontroller]** 그룹의 경우 이 그룹에 유효한 두 개의 **node_types** 를 할당할 수 있습니다. **node_type=control** 은 노드가 프로젝트 및 인벤토리 업데이트만 실행하지만 일반 작업은 실행할 수 없음을 나타냅니다. **node_type=hybrid** 에는 모든 것을 실행할 수 있습니다. 이 그룹의 기본값 = **하이브리드**. **[execution_nodes]** 그룹의 경우 이 그룹에 유효한 두 개의 node_types를 할당할 수 있습니다. **node_type=ECDHE** 는 노드가 실행 노드로 작업을 전달함을 나타냅니다. **node_type=execution** 은 노드에서 작업을 실행할 수 있음을 나타냅니다. 이 그룹의 기본값 = **실행**.

변수	설명
peer	*선택 사항* 피어 관계를 통해 노드 간 연결을 정의합니다. 이 변수는 다른 노드와의 네트워크 연결을 설정하는 데 사용되는 **receptor.conf** 파일에 **tcp-peer** 항목을 추가하는 데 사용됩니다. 피어링을참조하십시오. peers 변수는 인벤토리의 쉼표로 구분된 호스트 및/또는 그룹 목록일 수 있습니다. 이는 **receptor.conf** 파일을 구성하는 데 사용되는 호스트 세트로 확인됩니다.
pg_database	postgres 데이터베이스의 이름입니다. default = **awx**.
pg_host	외부에서 관리되는 데이터베이스일 수 있는 postgreSQL 호스트.
pg_password	postgreSQL 데이터베이스의 암호입니다. **pg_password** 에는 특수 문자를 사용하지 마십시오. 이로 인해 암호가 실패할 수 있습니다. 참고 PostgreSQL 13에서 더 이상 사용자 암호를 더 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	**prefer** 또는 **verify-full** 중 하나입니다. 클라이언트 측이 적용된 SSL의 **verify-full** 로 설정합니다. default = **prefer**.
pg_username	postgres 데이터베이스 사용자 이름입니다. default = **awx**.

변수	설명
postgres_ssl_cert	postgres ssl 인증서의 위치입니다. **/path/to/pgsql_ssl.cert**
`postgres_ssl_key	postgres ssl 키의 위치입니다. **/path/to/pgsql_ssl.key**
postgres_use_cert	postgres 사용자 인증서의 위치입니다. **/path/to/pgsql.crt**
postgres_use_key	postgres 사용자 키의 위치입니다. **/path/to/pgsql.key**
postgres_use_ssl	postgres가 SSL을 사용하는 경우
receptor_listener_port	recptor 연결에 사용할 포트입니다. 기본값은 27199입니다.
web_server_ssl_cert	*선택 사항* **/path/to/webserver.cert** **automationhub_ssl_cert** 와 동일하지만 웹 서버 UI 및 API의 경우입니다.
web_server_ssl_key	*선택 사항* **/path/to/webserver.key** **automationhub_server_ssl_key** 와 동일하지만 웹 서버 UI 및 API의 경우입니다.

A.6. ANSIBLE 변수

다음 변수는 **Ansible Automation Platform**이 원격 호스트와 상호 작용하는 방법을 제어합니다.

특정 플러그인 관련 변수에 대한 추가 정보는 **https://docs.ansible.com/ansible-core/devel/collections/ansible/builtin/index.html**에서 확인할 수 있습니다.

글로벌 구성 옵션 목록은 **https://docs.ansible.com/ansible-core/devel/reference_appendices/config.html**에서 확인할 수 있습니다.

변수	설명
ansible_connection	대상 호스트에서 작업에 사용되는 연결 플러그인입니다. 이는 ansible 연결 플러그인의 이름일 수 있습니다. SSH 프로토콜 유형은 **스마트**,**ssh** 또는 **paramiko** 입니다. 기본값 = **smart**
ansible_host	**inventory_hostname** 대신 사용할 대상 호스트의 IP 또는 이름입니다.
ansible_port	연결 포트 번호(예: SSH의 경우 22)입니다.
ansible_user	호스트에 연결할 때 사용할 사용자 이름입니다.
ansible_password	호스트에 인증하는 데 사용할 암호입니다. 이 변수는 일반 텍스트에 저장하지 않습니다. 항상 자격 증명 모음을 사용합니다.
ansible_ssh_private_key_file	ssh에서 사용하는 개인 키 파일입니다. 여러 키를 사용하고 SSH 에이전트를 사용하지 않는 경우 유용합니다.
ansible_ssh_common_args	이 설정은 항상 **sftp**,**scp**, **ssh** 의 기본 명령행에 추가됩니다. 특정 호스트(또는 그룹)에 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 파이프닝이 사용되는지 여부를 결정합니다. 이렇게 하면 **ansible.cfg** 에서 파이프링 설정을 덮어쓸 수 있습니다.
ansible_ssh_pass	
ansible_ssh_user	이 변수는 설치 프로그램의 SSH 사용자를 사용하도록 설정하고 기본값은 root로 설정합니다. 이 사용자는 암호 없이 SSH 기반 인증을 허용해야 합니다. SSH 키 기반 인증을 사용하는 경우 SSH 에이전트에서 키를 관리해야 합니다.

변수	설명
ansible_ssh_executable	(버전 2.2에 추가) 이 설정은 시스템 ssh를 사용하도록 기본 동작을 덮어씁니다. 그러면 **ansible.cfg** 에서 ssh_executable 설정을 덮어쓸 수 있습니다.
ansible_shell_type	대상 시스템의 쉘 유형입니다. **ansible_shell_executable** 을 비Bourne(sh) 호환 쉘로 설정하지 않는 한 이 설정을 사용해서는 안 됩니다. 기본적으로 명령은 sh 스타일 구문을 사용하여 포맷됩니다. 이 값을 **csh** 또는 **fish** 로 설정하면 대상 시스템에서 명령이 해당 쉘의 구문을 대신 따릅니다.
ansible_shell_executable	그러면 ansible 컨트롤러가 대상 시스템에서 사용하는 쉘을 설정하고 기본값인 **/bin/sh .cfg** 에서 실행 파일을 덮어씁니다. **/bin/sh** 를 사용할 수 없는 경우에만 변경되어야 합니다. 즉, 대상 시스템에 **/bin/sh** 가 설치되지 않았거나 sudo에서 실행할 수 없습니다.

다음 변수는 사용자가 직접 설정할 수 없습니다. Ansible은 항상 내부 상태를 반영하도록 이를 재정의합니다.

변수	설명
ansible_check_mode	확인 모드인지 여부를 나타내는 부울 값입니다.
ansible_dependent_role_names	현재 플레이에서 다른 플레이의 종속성으로 가져온 역할의 이름
ansible_limit	Ansible의 현재 실행을 위한 **--limit** CLI 옵션의 내용
ansible_loop	**loop_control.extended**를 사용하여 활성화된 경우 확장된 루프 정보가 포함된 사전 또는 맵
ansible_loop_var	loop_control.loop_var에 제공되는 값의 이름입니다. 2.8에 추가되었습니다.
ansible_index_var	**loop_control.index_var** 에 제공되는 값의 이름입니다. 2.9에 추가

변수	설명
ansible_parent_role_names	**include_role** 또는 **import_role** **작업** 에 의해 현재 역할을 실행하는 경우 이 변수에는 최근 역할이 있는 모든 상위 역할 목록이 포함됩니다. 즉, 이 역할을 포함하거나 가져온 역할은 목록의 첫 번째 항목입니다. 여러 포함이 발생하는 경우 이 목록의 첫 번째 항목은 마지막 역할입니다(이 역할을 포함하는 역할). 이 목록에 특정 역할이 두 번 이상 존재할 수도 있습니다. 예를 들어 역할 A에 역할 B가 포함된 경우 **ansible_parent_role_names** 는 ['A']와 동일합니다. 역할 B에 역할 C가 포함된 경우 목록은 ['B', 'A']가 됩니다.
ansible_parent_role_paths	**include_role** 또는 **import_role** 작업을 통해 현재 역할을 실행하는 경우 이 변수에는 최근 역할(즉, 이 역할을 포함/가입된 역할)이 목록의 첫 번째 요소인 모든 상위 역할 경로 목록이 포함됩니다. 이 목록에 있는 항목의 순서는 **ansible_parent_role_names** 를 참조하십시오.
ansible_play_batch	현재 플레이의 활성 호스트 목록은 직렬로 제한되는 aka **일괄** 처리로 실행됩니다. 실패 또는 연결할 수 없는 호스트는 **활성** 으로 간주되지 않습니다.
ansible_play_hosts	현재 플레이 실행의 호스트 목록은 직렬로 제한되지 않습니다. 실패하거나 연결할 수 없는 호스트는 이 목록에서 제외됩니다.
ansible_play_hosts_all	플레이의 대상인 모든 호스트 목록
ansible_play_role_names	현재 플레이로 가져온 역할의 이름입니다. 이 목록에는 종속성을 통해 암시적으로 포함된 역할 이름이 포함되어 있지 않습니다.
ansible_play_name	현재 실행된 플레이의 이름입니다. 2.8에 추가되었습니다. (플레이의 이름 특성, 플레이북의 파일 이름이 아닌)
ansible_search_path	즉, 작업 플러그인 및 조회에 대한 현재 검색 경로는 템플릿을 수행할 때 상대 경로를 검색합니다.
ansible_version	현재 실행 중인 ansible 버전에 대한 정보가 포함된 사전 또는 맵은 **전체**,**메이저**,**부**,**수정** 및 **문자열** 과 같은 키가 있습니다.