Red Hat Ansible Automation Platform 2.4

Red Hat OpenShift에 Ansible Automation Platform 2 배포

Last Updated: 2023-11-16

Red Hat Ansible Automation Platform 2.4 Red Hat OpenShift에 Ansible Automation Platform 2 배포

Roger Lopez
ansible-feedback@redhat.com

법적 공지

초록

이 문서에서는 Red Hat OpenShift에 Ansible Automation Platform 2를 배포하는 모범 사례를 설명합니다.

차례

의견 및 피드백

오픈 소스의 영점으로 Red Hat은 모든 사람이 참조 아키텍처에 대한 피드백과 의견을 제공하도록 초대합니다. 내부적으로 문서를 검토했지만 경우에 따라 문제가 발생하거나 오타 오류가 발생할 수 있습니다. 피드백은 당사가 생성하는 문서의 품질을 향상시킬 뿐만 아니라 독자가 잠재적인 개선 및 주제 확장에 대한 의견을 제공할 수 있도록 합니다. 문서에 대한 피드백은 ansible-feedback@redhat.com 로 이메일을 통해 제공될 수 있습니다. 이메일의 제목을 참조해 주십시오.

1장. 개요

Red Hat OpenShift 참조 아키텍처의 AAP(Ansible Automation Platform) 2.3은 Ansible Automation Platform 환경 배포에 대한 의견이 지정된 설정을 제공합니다. Ansible Automation Platform 2.3을 설치하고 구성하기 위한 최신 모범 사례에 따라 단계별 배포 절차를 제공합니다. Red Hat OpenShift에 Ansible Automation Platform을 배포하려는 시스템 및 플랫폼 관리자에게 가장 적합합니다.

Red Hat OpenShift의 기능을 활용하여 Ansible Automation Platform 배포를 간소화하고 설정하는 데 필요한 시간과 노력을 크게 줄일 수 있습니다.

그림 1.1. 자동화 컨트롤러 아키텍처

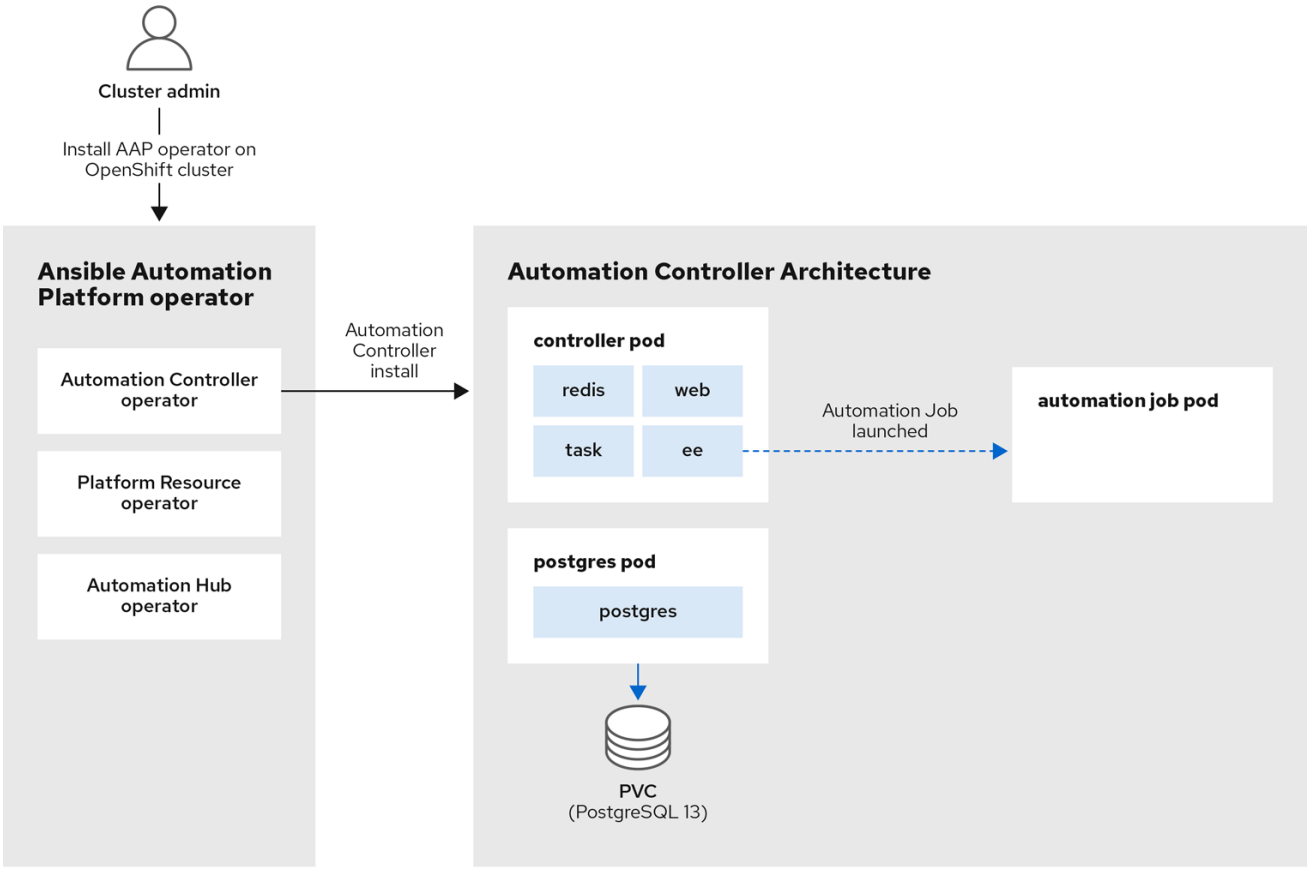
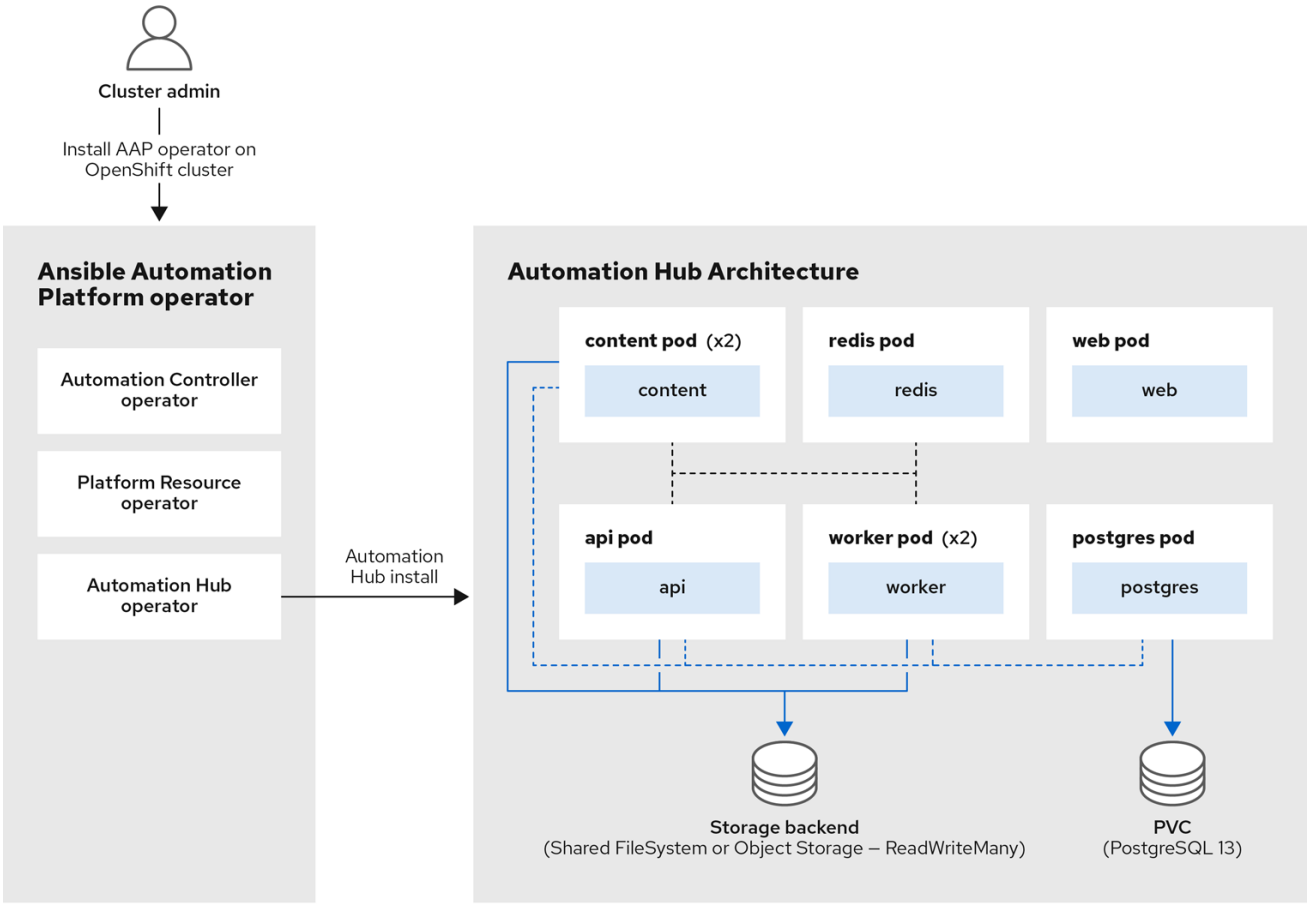


그림 1.1. "자동화 컨트롤러 아키텍처" 는 자동화 컨트롤러 구성 요소를 배포하는 AAP(Ansible Automation Platform) Operator의 배포 프로세스 흐름을 보여줍니다. 대규모 Ansible Automation Platform Operator를 구성하는 세 개의 Operator 중 하나인 자동화 컨트롤러 Operator는 컨트롤러, postgres 및 Automation 작업 Pod를 포함하여 다양한 Pod를 배포합니다.

그림 1.2. 자동화 허브 아키텍처



마찬가지로 그림 1.2. "자동화 허브 아키텍처" 에는 자동화 허브 구성 요소를 배포하는 AAP Operator가 표시됩니다. 자동화 허브 Operator는 서로 통신하는 다양한 포드를 배포하여 내부적으로 생성된 콘텐츠, Red Hat Ansible 인증 콘텐츠, 실행 환경 및 Ansible 검증 콘텐츠를 팀과 공유하는 자동화 허브를 제공합니다.

또한 이 참조 아키텍처에서는 자동화 노력을 위한 견고한 기반이 제공하는 효율적이고 확장 가능한 환경을 제공하는 데 관련된 주요 단계를 중점적으로 설명합니다.

2장. RED HAT OPENSHIFT에서 ANSIBLE AUTOMATION PLATFORM을 사용하는 이유는 무엇입니까?

Red Hat OpenShift 4.x에서 Ansible Automation Platform 2.3을 실행하면 플랫폼을 배포 및 관리하는 보다 효율적이고 자동화된 방법뿐만 아니라 Red Hat OpenShift의 기본 모니터링, 로깅, 보안 및 확장성 기능과 보다 효과적으로 통합할 수 있습니다.

또한 Red Hat OpenShift의 OLM(Operator Lifecycle Manager)을 사용하여 Ansible Automation Platform Operator를 배포하고 관리하면 업데이트 및 업그레이드 프로세스를 단순화하여 제어 및 유연성을 높일 수 있습니다.

이러한 혜택에 대한 추가 분석 정보는 아래에서 확인할 수 있습니다.

- **자동화:** Ansible Automation Platform Operator는 Red Hat OpenShift에서 Ansible Automation Platform의 배포 및 관리를 자동화하는 방법을 제공합니다. 이를 통해 플랫폼을 배포 및 관리하는 데 필요한 시간과 노력을 줄이고 일관되고 예측 가능한 방식으로 실행되고 있는지 확인할 수 있습니다.

- **확장성:** Operator는 조직의 요구 사항을 충족하도록 Ansible Automation Platform을 확장하는 데 도움이 될 수 있습니다. 플랫폼 인스턴스를 여러 개 배포하고 단일 위치에서 관리할 수 있습니다.

- **유연성:** Operator는 Ansible Automation Platform을 배포하고 관리하는 유연한 방법을 제공합니다. 특정 요구 사항에 맞게 플랫폼 구성을 사용자 지정하고 개발, 스테이징 및 프로덕션과 같은 다양한 환경에 배포할 수 있습니다.

- **모니터링 및 문제 해결:** Operator는 Prometheus 및 Grafana와 같은 Red Hat OpenShift의 내장 모니터링 및 로깅 툴과 통합되며 플랫폼의 리소스 사용량을 모니터링하고 병목 현상을 식별하는 데 사용할 수 있습니다.

- **관리 및 업그레이드:** Red Hat OpenShift 4.x와 함께 제공되는 OLM(Operator Lifecycle Manager)을 사용하면 쉽게 배포, 관리 및 업그레이드할 수 있습니다.[1] 클러스터 전체의 Ansible Automation Platform Operator입니다.

---

[1] Ansible Automation Platform 지원 라이프사이클 버전 - https://access.redhat.com/support/policy/updates/ansible-automation-platform

3장. 시작하기 전

Red Hat OpenShift에 AAP를 배포하기 전에 설치 전에 해결해야 할 주요 고려 사항을 이해하는 것이 중요합니다. 이러한 요인은 라이프사이클 전반에 걸쳐 AAP 환경의 상태 및 확장성을 결정합니다.

이 섹션에서는 다음과 같은 주요 고려 사항에 대한 분석을 찾을 수 있습니다.

- Red Hat OpenShift 리소스 관리
- 자동화 컨트롤러 Pod 컨테이너의 크기 조정 권장 사항
- Postgres Pod의 크기 조정 권장 사항
- 자동화 작업 Pod의 크기 조정 권장 사항
- 자동화 허브 Pod의 크기 조정 권장 사항

성공적인 배포의 주요 측면 중 하나는 Red Hat OpenShift 클러스터에 대한 AAP 애플리케이션의 최적 성능과 가용성을 보장하기 위한 Pod 및 컨테이너의 적절한 리소스 관리입니다.

3.1. POD 및 컨테이너의 리소스 관리

리소스 관리와 관련된 두 가지 주요 리소스는 CPU 및 메모리(RAM)입니다. Red Hat OpenShift는 리소스 요청 및 리소스 제한을 사용하여 컨테이너에서 Pod에서 사용할 수 있는 리소스의 양을 제어합니다.

3.1.1. 리소스 요청이란 무엇입니까?

리소스 요청은 컨테이너가 제대로 실행되고 작동하는 데 필요한 최소한의 리소스 양입니다. Kubernetes 스케줄러는 이 값을 사용하여 컨테이너에 사용 가능한 리소스가 충분한지 확인합니다.

3.1.2. 리소스 제한이란 무엇입니까?

반면 리소스 제한은 컨테이너에서 사용할 수 있는 최대 리소스 양입니다. 리소스 제한을 설정하면 컨테이너에서 필요한 것보다 더 많은 리소스를 사용하지 않으므로 다른 컨테이너가 리소스 중단으로 어려움을 겪을 수 있습니다.

3.1.3. 리소스 관리가 중요한 이유는 무엇입니까?

AAP는 올바른 리소스 요청 및 제한을 설정하는 것이 중요합니다. 리소스 할당이 부적절하면 제어 Pod가 종료되어 자동화 컨트롤러 내에서 모든 자동화 작업이 손실될 수 있습니다.

3.1.4. 리소스 계획

적절한 리소스 관리 값을 설정하는 동안 조직에서는 사용 가능한 리소스에 따라 요구 사항에 가장 적합한 아키텍처를 고려해야 합니다. 예를 들어 Ansible Automation Platform 환경의 고가용성이 자동화 작업을 실행할 수 있는 용량을 극대화하는 것보다 더 중요한지 여부를 결정합니다.

자세한 내용은 이 참조 아키텍처에 사용된 기존 Red Hat OpenShift 환경을 살펴보겠습니다. 다음으로 구성됩니다.

- 컨트롤 플레인 노드 세 개
- 작업자 노드 3개

이러한 각 노드는 4개의 vCPU와 16GiB의 RAM으로 구성됩니다.

Red Hat OpenShift 클러스터의 컨트롤 플레인 노드는 애플리케이션을 실행하지 않으므로 사용 가능한 3개의 작업자 노드에 중점을 둡니다.

이러한 3개의 작업자 노드를 사용하면 Ansible Automation Platform 가용성을 극대화하거나 최대한 많은 자동화 작업을 실행하는 등 더 중요한 것이 무엇인지 확인해야 합니다.

가용성이 가장 중요한 경우 두 컨트롤 Pod가 별도의 작업자 노드(예: **worker0** 및 **worker**1)에서 실행되는 반면 모든 자동화 작업은 나머지 작업자 노드(예: worker2) 내에서 실행되도록 하는 데 중점을 둡니다.

그러나 이렇게 하면 자동화 작업에 사용할 수 있는 리소스가 절반으로 줄어들어 제어 Pod와 자동화 Pod를 동일한 작업자 노드에서 실행하지 않는 것이 좋습니다.

실행할 자동화 작업의 양을 극대화하는 것이 주요 목표인 경우, 컨트롤 Pod에 하나의 작업자 노드(예: worker0)를 사용하고 자동화 작업을 실행하는 데 나머지 두 개의 작업자 노드(예: worker1 및 worker2)를 사용하면 제어 Pod의 중복성이 없는 경우 작업의 실행 가능한 리소스가 두 배로 증가합니다.

물론 이 솔루션은 둘 다 동일하게 중요할 수 있으며, 이 경우 두 요구 사항을 충족하기 위해 추가 리소스(예: 더 많은 작업자 노드 추가)가 필요합니다.

3.2. 자동화 컨트롤러 POD 컨테이너의 크기 권장 사항

개요 섹션에서 그림 1.1. "자동화 컨트롤러 아키텍처" 를 다시 보면 제어 Pod에 4개의 컨테이너가 포함되어 있습니다.

- **web**
- **ee**
- **Redis**
- **task**

이러한 각 컨테이너는 Ansible 자동화 컨트롤러에서 고유한 기능을 수행하고 리소스 구성이 제어 Pod에 미치는 영향을 이해하는 것이 중요합니다. 기본적으로 Red Hat OpenShift는 최소 테스트 설치에 충분하지만 프로덕션에서 Ansible Automation Platform을 실행하는 데는 적합하지 않습니다.

Ansible Automation Platform의 Red Hat OpenShift 기본값은 다음과 같습니다.

- CPU: 100m
- 메모리: 128Mi

기본적으로 Red Hat OpenShift는 최대 리소스 제한을 구성하지 않으며 Ansible Automation Platform 제어 Pod에서 요청하는 모든 가능한 리소스를 할당하려고 합니다. 이 구성으로 인해 리소스가 중단되고 Red Hat OpenShift 클러스터에서 실행되는 다른 애플리케이션에 영향을 미칠 수 있습니다.

제어 Pod에서 컨테이너의 리소스 요청 및 제한에 대한 시작점을 설명하기 위해 다음 가정을 사용합니다.

- Red Hat OpenShift 클러스터 내에서 각각 4개의 vCPU 및 16GiB RAM이 있는 작업자 노드 3개
- 자동화 작업을 위한 리소스 극대화가 고가용성보다 더 중요합니다.
- 자동화 컨트롤러 실행을 위한 전용 작업자 노드 1개
- 자동화 작업을 실행하기 위한 나머지 두 개의 작업자 노드

컨트롤 Pod 내에서 컨테이너의 크기를 조정하는 경우 워크로드의 세부 사항을 고려하는 것이 중요합니다. 이 참조 환경에 대한 특정 권장 사항을 제공하는 성능 테스트를 수행했지만 이러한 권장 사항은 모든 유형의 워크로드에 적용되지 않을 수 있습니다.

시작점으로 성능 컬렉션 플레이북, 특히 chatty_tasks.yml 을 활용하기로 결정했습니다.

성능 벤치마크는 다음과 같이 구성되어 있습니다.

- 호스트 1개로 인벤토리 생성

- **chatty _tasks.yml** 파일을 실행하는 작업 템플릿 생성

chatty _tasks 작업 템플릿은 **ansible.builtin.debug** 모듈을 사용하여 호스트당 일련의 디버그 메시지를 생성하고 필요한 인벤토리를 생성합니다. **ansible.builtin.debug** 모듈을 사용하면 추가 오버헤드를 도입하지 않고도 자동화 컨트롤러의 성능을 정확하게 표시할 수 있습니다.

작업 템플릿은 **10**에서 **50** 사이의 지정된 동시성 수준으로 실행되었으며 작업 템플릿의 동시 호출 수를 나타냅니다.

아래에 표시된 리소스 요청 및 리소스 제한은 성능 벤치마크의 결과이며 유사한 리소스가 있는 **Red Hat OpenShift** 클러스터에서 **AAP**를 실행하기 위한 시작 기준으로 사용할 수 있습니다.

```
spec:
...
  ee_resource_requirements:
    limits:
      cpu: 500m
      memory: 400Mi
    requests:
      cpu: 100m
      memory: 400Mi
  task_resource_requirements:
    limits:
      cpu: 4000m
      memory: 8Gi
    requests:
      cpu: 1000m
      memory: 8Gi
  web_resource_requirements:
    limits:
      cpu: 2000m
      memory: 1.5Gi
    requests:
      cpu: 500m
      memory: 1.5Gi
  redis_resource_requirements:
    limits:
      cpu: 500m
      memory: 1.5Gi
    requests:
      cpu: 250m
      memory: 1.5Gi
```

참고

Red Hat OpenShift 클러스터 내에서 메모리 리소스를 과도하게 사용하지 않도록 메모리 리소스 요청 및 제한이 일치하므로 Pod가 OOM(Out Of Memory)이 종료될 수 있습니다. 리소스 제한이 리소스 요청보다 크면 Red Hat OpenShift 노드를 과도하게 사용할 수 있는 시나리오가 발생할 수 있습니다.

참고

CPU 리소스가 압축 가능한 것으로 간주되므로 CPU 리소스 요청 및 제한은 메모리와 다릅니다. 즉, Red Hat OpenShift는 리소스 제한에 도달할 때 컨테이너의 CPU를 제한하려고 하지만 컨테이너를 종료하지는 않습니다. 제어 Pod 내의 위의 컨테이너에서는 지정된 워크로드에 충분한 CPU가 제공되지만 임계값(CPU 제한)을 더 높은 값으로 설정하여 부하에서 더 높은 CPU 요청이 제공되었습니다.

주의

위의 시나리오는 다른 애플리케이션이 해당 작업자 노드 내의 리소스를 사용하지 않고 제어 Pod가 전용 Red Hat OpenShift 작업자 노드를 사용하므로 이 노드에 상주하지 않는 것으로 가정하고 있습니다. 자세한 내용은 3.7절. “자동화 컨트롤러 Pod를 위한 전용 노드 지정” 에서 확인할 수 있습니다.

3.3. POSTGRES POD의 크기 권장 사항

chatty_task 플레이북을 사용하여 성능 벤치마크 테스트를 수행한 후 500m 미만의 CPU 리소스 요청으로 인해 초기 리소스 요청 이상으로 요청된 추가 리소스가 Pod에서 보장되지 않으므로 500m 미만의 CPU 리소스 요청으로 인해 Postgres Pod에서 CPU 리소스가 제한될 수 있습니다. 그러나 500m 요청을 초과하는 테스트 중에 버스트가 있기 때문에 CPU 제한이 1000m(1 vCPU)으로 설정되었습니다.

메모리는 압축할 수 없는 리소스가 아니므로 chatty_task 성능에서는 테스트에서 가장 높은 수준의 Postgres Pod가 650Mi 이상의 RAM을 사용하는 것으로 확인되었습니다.

따라서 결과에 따라 이 참조 환경에 대한 메모리 리소스 요청 및 제한 권장 사항은 충분한 버퍼를 제공하고 Postgres Pod의 잠재적인 OOM(Out Of Memory) 종료를 방지하기 위해 1Gi입니다.

아래에 표시된 리소스 요청 및 리소스 제한은 성능 벤치마크 테스트의 결과이며 Postgres Pod를 실행하기 위한 시작 기준으로 사용할 수 있습니다.

```
spec:
...
  postgres_resource_requirements:
    limits:
      cpu: 1000m
      memory: 1Gi
    requests:
      cpu: 500m
      memory: 1Gi
```

주의

아래 값은 이 참조 환경에 고유하며 워크로드에는 충분하지 않을 수 있습니다. **Postgres** 포드의 성능을 모니터링하고 성능 요구 사항에 맞게 리소스 할당을 조정하는 것이 중요합니다.

3.4. 자동화 작업 POD의 크기 권장 사항

Ansible Automation Platform 작업은 호스트 인벤토리에 대해 **Ansible** 플레이북을 시작하는 자동화 컨트롤러의 인스턴스입니다. **Ansible Automation Platform**이 **Red Hat OpenShift**에서 실행되는 경우 기본 실행 대기열은 설치 시 **Operator**가 생성한 컨테이너 그룹입니다.

컨테이너 그룹은 **Kubernetes** 인증 정보 및 기본 **Pod** 사양으로 구성됩니다. 컨테이너 그룹으로 작업이 시작되면 컨테이너 그룹 **Pod** 사양에 지정된 네임스페이스의 자동화 컨트롤러에서 **Pod**가 생성됩니다. 이러한 **Pod**를 자동화 작업 **Pod**라고 합니다.

자동화 작업 **Pod**에 대한 적절한 크기를 결정하려면 먼저 자동화 컨트롤러 컨트롤 플레인에서 동시에 시작할 수 있는 작업 수의 기능을 이해해야 합니다.

이 예에서는 작업자 노드(각 **4 vCPU** 및 **16GiB**의 **RAM**)가 **3**개 있습니다. 하나의 작업자 노드는 제어 **Pod**를 호스팅하고 나머지 두 개의 작업자 노드는 자동화 작업에 사용됩니다.

이러한 값을 기반으로 자동화 컨트롤러 컨트롤 플레인을 실행할 수 있는 컨트롤 용량을 결정할 수 있습니다.

다음 공식은 분석을 제공합니다.

총 제어 용량 = MB 단위의 총 메모리 / 포크 크기

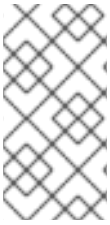
작업자 노드에 따라 다음과 같이 표시할 수 있습니다.

총 제어 용량 = 16,000MB / 100MB = 160

참고

계산에 대한 자세한 내용은 용량 알고리즘에 대한 리소스 결정을 검토하십시오.

즉, 자동화 컨트롤러가 160개의 작업을 동시에 시작하도록 구성되어 있습니다. 그러나 컨테이너 그룹/실행 플레인 용량과 일치하도록 이 값을 조정해야 합니다.



참고

단순화를 위해 16GB는 16,000MB로 반올림되며 하나의 포크 크기는 기본적으로 100MB입니다.

이제 사용 가능한 제어 용량을 계산했으므로 최대 동시 자동화 작업 수를 확인할 수 있습니다.

이를 확인하려면 컨테이너 그룹/실행 플레인 내의 자동화 작업 Pod 사양에 vCPU 250m 및 100Mi of RAM의 기본 요청이 있음을 알고 있어야 합니다.

하나의 작업자 노드의 총 메모리를 사용합니다.

16,000MB / 100MiB = 160 동시 작업

하나의 작업자 노드의 총 CPU 사용

4000밀리CPU / 250밀리cpu = 16 동시 작업

위의 값을 기반으로 노드의 최대 동시 작업 수를 두 개의 동시 작업 값 **- 16** 중 가장 작은 값으로 설정해야 합니다. 예제에서는 자동화 작업을 실행하는 데 할당된 작업자 노드가 두 개이므로 이 숫자는 작업자 노드당 동시 작업 **16**개입니다.

자동화 컨트롤러의 구성은 현재 **160**개의 동시 작업으로 설정되었으며 사용 가능한 작업자 노드 용량은 **32**개의 동시 작업만 허용합니다. 이 문제는 숫자가 균형을 해제하기 때문에 발생합니다.

즉, 자동화 컨트롤러의 컨트롤 플레인은 **160**개의 작업을 동시에 시작할 수 있다고 생각하지만 **Kubernetes** 스케줄러는 컨테이너 그룹 네임스페이스에서 동시에 최대 **32**개의 자동화 작업 **Pod**만 스케줄링합니다.

컨트롤 플레인과 컨테이너 그룹/실행 플레인 간의 균형을 해제하면 다음과 같은 문제가 발생할 수 있습니다.

- 컨트롤 플레인 용량이 컨테이너 그룹의 최대 작업 **Pod** 수보다 크면 컨트롤 플레인에서 시작할 **Pod**를 제출하여 작업을 시작합니다. 그러나 이러한 **Pod**는 리소스를 사용할 수 있게 될 때까지 실제로 실행되지 않습니다. 작업 **Pod**가 **AWX_CONTAINER_GROUP_POD_PENDING_TIMEOUT** 의 시간 초과 내에서 시작되지 않으면 작업이 중단됩니다(기본값은 **2**시간).

- 컨테이너 그룹이 컨트롤 플레인에서 시작할 수 있다고 생각한 것보다 더 많은 동시 자동화 작업을 지원할 수 있는 경우, 컨테이너 그룹이 지원할 수 있는 최대 동시 자동화 작업에 도달하기 위해 자동화 컨트롤러에서 충분한 자동화 작업을 시작하지 않으므로 이 용량이 효과적으로 소모됩니다.

중단된 작업 또는 사용되지 않는 리소스를 방지하려면 유효 제어 용량을 기본 컨테이너 그룹에서 지원할 수 있는 최대 동시 작업 수와 균형을 유지하는 것이 좋습니다.

컨트롤 플레인이 시작할 최대 작업 수는 **AWX_CONTROL_NODE_TASK_IMPACT** 설정의 영향을 받기 때문에 "효율 제어 용량"이라는 용어를 사용합니다. **AWX_CONTROL_NODE_TASK_IMPACT** 변수는 자동화 작업별로 제어 **Pod**에서 사용할 수 있는 용량을 정의하여 제어 **Pod**가 시작하려고 하는 자동화 작업 수를 효과적으로 제어합니다.

효과적인 제어 용량과 사용 가능한 실행 용량 간의 균형을 유지하기 위해 **AWX_CONTROL_NODE_TASK_IMPACT** 변수를 자동화 컨트롤러 컨트롤 플레인에서 실행할 동시 작업

수를 컨테이너 그룹/실행 플레인에서 시작할 자동화 작업 Pod 수와 일치하도록 제한하는 값으로 설정할 수 있습니다.

컨테이너 그룹이 지원할 수 있는 것보다 더 많은 동시 자동화 작업을 시작하지 않도록 AWX_CONTROL_NODE_TASK_IMPACT 의 최적 값을 계산하려면 다음 공식을 사용하면 됩니다.

AWX_CONTROL_NODE_TASK_IMPACT = 컨테이너 그룹에서 시작할 수 있는 최대 동시 작업/제어 용량 / 최대 동시 작업

참조 환경의 경우 다음과 같습니다.

AWX_CONTROL_NODE_TASK_IMPACT = 160 / 32 = 5

따라서 이 참조 환경에서는 AWX_CONTROL_NODE_TASK_IMPACT 가 5와 같아야 합니다. 이 값은 6장. *자동화 컨트롤러 설치* 장의 extra_setting 부분 내에 설정되며 이 문서의 뒷부분에서 다룹니다.

3.5. 자동화 컨트롤러 POD 크기 권장 사항 요약

컨트롤 플레인(컨트롤 Pod)의 리소스 요청 및 제한을 적절히 설정하고 컨트롤 및 실행 용량의 균형을 유지하기 위해 컨테이너 그룹/실행 플레인(자동화 작업 Pod)을 적절히 설정해야 합니다. 다음 사항에 따라 올바른 구성을 결정할 수 있습니다.

- 제어 용량 계산
- 동시에 실행할 수 있는 자동화 작업 수 계산
- 자동화 컨트롤러 설치 내에서 적절한 균형 값으로 AWX_CONTROL_NODE_TASK_IMPACT 변수 설정

3.6. 자동화 허브 POD의 크기 권장 사항

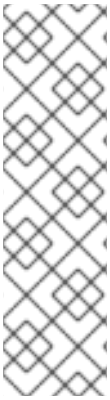
개요 섹션에 설명된 그림 1.2. “자동화 허브 아키텍처” 에서는 배포가 각각 컨테이너를 호스팅하는 7개의 Pod로 구성됩니다.

Pod 목록은 다음으로 구성됩니다.

- 콘텐츠**(x2)**
- **Redis**
- **api**
- **Postgres**
- 작업자 **(x2)**

자동화 허브 아키텍처를 구성하는 **7**개의 **Pod**는 함께 협력하여 컨텐츠를 효율적으로 관리 및 배포할 수 있으며 자동화 허브 환경의 전반적인 성능 및 확장성에 매우 중요합니다.

이러한 **Pod** 중 작업자 **Pod**는 콘텐츠를 처리, 동기화 및 배포를 수행하기 때문에 특히 중요합니다. 이로 인해 적절한 리소스 양을 작업자 **Pod**로 설정하여 작업을 수행할 수 있도록 하는 것이 중요합니다.



참고

다음은 자동화 허브 환경에 필요한 리소스 요청 및 제한에 대한 추정치를 제공하기 위한 지침입니다. 실제 리소스 요구 사항은 설정에 따라 달라집니다.

예를 들어 자주 업데이트 또는 동기화를 수행하는 리포지토리 수가 많은 환경에는 처리 부하를 처리하기 위해 더 많은 리소스가 필요할 수 있습니다.

이 참조 환경에서는 **Pod** 크기를 결정하기 위해 원격 리포지토리의 자동화 허브 환경에서 수행할 수 있는 가장 높은 메모리 사용 작업 중 하나를 사용하여 사전 테스트를 수행했습니다.

결과에 따라 자동화 허브 내에서 원격 리포지토리를 성공적으로 동기화하려면 각 **Pod**에 대해 다음 리소스 요청 및 리소스 제한을 설정해야 합니다.

```
spec:
...
content:
  resource_requirements:
    limits:
      cpu: 250mm
      memory: 400Mi
    requests:
      cpu: 100m
      memory: 400Mi

redis:
  resource_requirements:
    limits:
      cpu: 250m
      memory: 200Mi
    requests:
      cpu: 100m
      memory: 200Mi

api:
  resource_requirements:
    limits:
      cpu: 250m
      memory: 400Mi
    requests:
      cpu: 150m
      memory: 400Mi

postgres_resource_requirements:
  resource_requirements:
    limits:
      cpu: 500m
      memory: 1Gi
    requests:
      cpu: 200m
      memory: 1Gi

worker:
  resource_requirements:
    limits:
      cpu: 1000m
      memory: 3Gi
    requests:
      cpu: 400m
      memory: 3Gi
```

3.7. 자동화 컨트롤러 POD를 위한 전용 노드 지정

제어 **Pod**와 자동화 작업 **Pod**를 분리하고 이러한 두 유형의 **Pod** 간 리소스 경합을 방지하려면 전용 노드에서 제어 **Pod**를 실행하는 것이 중요합니다**.** 이러한 분리는 리소스 제약 조건으로 인한 성능 저하의 위험 없이 제어 **Pod** 및 제공하는 서비스의 안정성과 신뢰성을 유지하는 데 도움이 됩니다**.**

이 참조 환경에서는 실행할 수 있는 자동화 작업 수를 극대화하는 데 중점을 둡니다. 즉, **Red Hat OpenShift** 환경에서 사용 가능한 **3**개의 작업자 노드 중 하나의 작업자 노드는 제어 **Pod**를 실행하는 데 전용되지만 나머지 **2**개의 작업자 노드는 자동화 작업 실행에 사용됩니다.

주의

제어 **Pod**를 실행하기 위해 하나의 작업자 노드만 노력하면 전용 작업자 노드가 중단된 경우 다른 위치에 있지 않기 때문에 서비스가 손실될 위험이 발생할 수 있습니다. 이러한 상황을 해결하려면 **Red Hat OpenShift** 클러스터 내에서 추가 제어 **Pod** 복제본을 실행하기 위해 자동화 작업을 실행하거나 추가 작업자 노드를 추가하는 작업자 노드 수를 줄이는 것은 실행 가능한 옵션입니다.

3.7.1. 자동화 컨트롤러의 특정 작업자 노드에 제어 Pod 할당

Red Hat OpenShift의 특정 노드에 제어 **Pod**를 할당하려면 **Pod** 사양에서 **node_selector** 필드와 **topology_spread_constraints** 필드를 사용하는 조합이 사용됩니다. **node_selector** 필드를 사용하면 **Pod**를 호스팅할 수 있도록 노드가 일치해야 하는 라벨 기준을 지정할 수 있습니다. 예를 들어 **aap_node_type: control** 라벨이 있는 노드가 있는 경우 **Pod** 사양에 다음을 지정하여 이 노드에 **Pod**를 할당합니다.

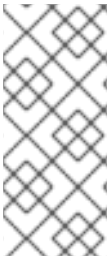
```
spec:
...
  node_selector: |
    aap_node_type: control
```

topology_spread_constraints 는 **aap_node_type: control** 레이블이 **1**인 노드에서 예약할 수 있는 최대 **Pod** 수**(maxSkew)**를 설정합니다. **topologyKey** 는 노드의 호스트 이름을 나타내는 기본 제공 레이블인 **kubernetes.io/hostname** 로 설정됩니다. **whenUnsatisfiable** 설정이 **ScheduleAnyway** 로 설정되어 제약 조건을 충족하는 데 필요한 레이블이 있는 노드가 충분하지 않은 경우 **Pod**를 예약할 수 있습니다. **labelSelector** 는 **aap_node_type: control** 레이블이 있는 **Pod**와 일치합니다. 이 문제의 영향은 **Red Hat OpenShift**가 노드당 단일 컨트롤러 **Pod** 예약을 우선시한다는 것입니다. 그러나 사용 가능한 작업자 노드보다 복제본 요청이 더 많은 경우 **Red Hat OpenShift**를 사용하면 리소스가 충분한 경우 동일한 기존 작업자 노드에서 여러 컨트롤러 **Pod**를 예약할 수 있습니다.

tolerations 섹션은 라벨이 **dedicated: AutomationController** 인 노드에서만 **Pod**를 예약할 수 있도록 지정합니다. 허용 오차의 영향은 **NoSchedule** 로 설정되어 필수 라벨이 없는 노드에 **Pod**를 예약하지 않도록 합니다. 이는 **topology_spread_contstraints** 와 결합하여 **Pod**를 노드에 분배하는 방법뿐만 아니라 예약할 수 있는 노드를 나타내는 데 사용됩니다.

```
spec:
```

```
...
  topology_spread_constraints: |
    - maxSkew: 1
      topologyKey: "kubernetes.io/hostname"
      whenUnsatisfiable: "ScheduleAnyway"
      labelSelector:
        matchLabels:
          aap_node_type: control
  tolerations: |
    - key: "dedicated"
      operator: "Equal"
      value: "AutomationController"
      effect: "NoSchedule"
```



참고

노드 레이블 및 테인트의 애플리케이션은 부록 C. *Red Hat OpenShift 노드에 레이블 및 테인트 적용* 내에서 확인할 수 있습니다. 사양 파일에 노드 선택기, 토폴로지 제약 조건 및 허용 오차를 추가하는 단계는 6장. *자동화 컨트롤러 설치* 에 표시됩니다.

3.8. 데이터베이스 고가용성 처리

Ansible Automation Platform 내에 자동화 컨트롤러 및 자동화 허브 구성 요소를 배포하면 **PostgreSQL** 데이터베이스의 **PVC**를 활용할 수 있습니다. 이러한 **PVC**의 가용성이 **Ansible Automation Platform** 실행의 안정성에 중요합니다.

Postgres Operator(PGO) 및 **ODF(OpenShift Data Foundation)**를 통해 **Crunchy Data**에서 제공하는 것과 같이 **Red Hat OpenShift** 클러스터 내에서 **PVC** 가용성을 처리하는 데 사용할 수 있는 몇 가지 전략이 있습니다.

Cunchy Data는 **PostgreSQL** 클러스터를 자동으로 관리하는 선언적 **Postgres** 솔루션을 제공하는 **Postgres Operator**인 **PGO**를 제공합니다. **PGO**를 사용하면 **Postgres** 클러스터를 생성하고, **HA(**고가용성**) Postgres** 클러스터를 생성하고 이를 **Ansible Automation Platform**과 같은 애플리케이션에 연결할 수 있습니다.

OpenShift Data Foundation(ODF)은 컨테이너화된 애플리케이션의 영구 스토리지를 관리할 수 있는 고가용성 스토리지 솔루션입니다. **Ceph, NooBaa** 및 **Rook**을 포함한 여러 오픈 소스 운영자 및 기술로 구성됩니다. 이러한 다양한 **Operator**를 사용하면 **Ansible Automation Platform**과 같은 애플리케이션에 연결할 수 있는 파일, 블록 및 오브젝트 스토리지를 프로비저닝하고 관리할 수 있습니다.

참고

PostgreSQL 데이터베이스에 고가용성 **PVC**를 제공하는 단계는 이 참조 아키텍처의 범위를 벗어납니다.

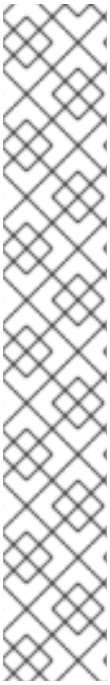
4장. 사전 요구 사항

이 참조 환경에 **Ansible Automation Platform 2.3**을 설치하면 다음이 사용됩니다.

- **Red Hat OpenShift Platform 4.12**
- 부록 C. *Red Hat OpenShift 노드에 레이블 및 테인트 적용*
- 자동화 허브에 대한 **ReadWriteMany** 스토리지 요구 사항을 처리하는 **Amazon S3** 버킷

주의

AAP 2.3을 설치하려면 **Red Hat OpenShift 4.9** 이상 버전이 필요합니다. 자세한 내용은 **Red Hat Ansible Automation Platform** 라이프 사이클 페이지를 참조하십시오.

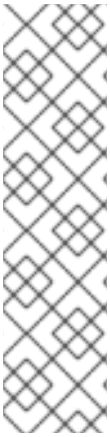


참고

Red Hat OpenShift를 배포하는 방법에는 여러 가지가 있습니다. **Red Hat OpenShift** 클러스터의 크기는 애플리케이션의 특정 요구 사항(**Ansible Automation Platform**뿐만 아니라)에 따라 달라집니다. 고려해야 할 요소에는 클러스터에 액세스하는 사용자 수, 애플리케이션이 실행하는 데 필요한 데이터 및 리소스 양, 확장성 및 중복 요구 사항이 포함됩니다.

Red Hat OpenShift를 배포하는 방법에 대한 자세한 내용은 설치 가이드를 참조하십시오.

위의 설치 가이드 외에도 **OpenShift 4 Resources Configuration: Methodology** 및 **Tools** 블로그 문서를 참조하여 필요에 따라 적절한 클러스터 크기를 결정합니다.



참고

자동화 허브에는 여러 **Pod**가 컬렉션과 같은 공유 콘텐츠에 액세스할 수 있도록 작업을 위해 **ReadWriteMany** 파일 기반 스토리지, **Azure Blob** 스토리지 또는 **Amazon S3** 호환 스토리지가 필요합니다.

이 참조 환경은 **ReadWriteMany** 스토리지 요청을 충족하기 위해 **Amazon S3** 버킷을 생성합니다. 세부 정보 부록 D. *Amazon S3 버킷 생성*.

5장. ANSIBLE AUTOMATION PLATFORM OPERATOR 설치

Ansible Automation Platform Operator를 설치할 때 권장되는 배포 방법은 수동 업데이트 승인을 사용하여 대상 네임스페이스에 클러스터 범위 Operator를 설치하는 것입니다.

이 배포 방법의 주요 장점은 다른 네임스페이스를 통해 처리되는 방식에서 AAP Operator의 범위를 제한하려는 경우 유연성을 제공할 수 있는 대상 네임스페이스 내의 리소스에만 영향을 미치는 것입니다.

예를 들어 업그레이드를 테스트하는 동안 다른 AAP 배포를 관리하기 위해 별도의 *devel* 및 *prod* 네임스페이스가 있어야 합니다.

Ansible Automation Platform Operator를 배포하는 단계는 다음과 같습니다.

- 클러스터 인증 정보를 사용하여 Red Hat OpenShift 웹 콘솔에 로그인합니다.
- 왼쪽 탐색 메뉴에서 Operator → OperatorHub를 선택합니다.
- Ansible Automation Platform을 검색하고 선택합니다.
- Ansible Automation Platform 설치 페이지에서 "설치"를 선택합니다.
- "Install Operator" 페이지에서,
  - 적절한 업데이트 채널, stable-2.3-cluster-scoped선택
  - 적절한 설치 모드, 클러스터의 특정 네임스페이스를선택합니다.
  - 적절한 설치된 네임스페이스, Operator 권장 Namespace: aap을선택합니다.
  - 적절한 업데이트 승인을 선택합니다(예: Manual).

- 설치를 클릭합니다.

- 필요한 수동 승인 에서 승인을 클릭합니다.

Ansible Automation Platform을 설치하는 프로세스는 사용할 수 있을 때까지 몇 분 정도 걸릴 수 있습니다.

설치가 완료되면 **Operator** 보기 버튼을 선택하여 설치 중에 지정된 네임스페이스에 설치된 **Operator**를 확인합니다(예: **aap**).

참고

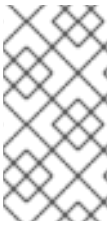
이 **AAP Operator** 배포는 네임스페이스 **aap** 만 대상으로 합니다. **AAP Operator**에 의해 추가 네임스페이스를 대상으로 하는 경우 **OperatorGroup** 사양 파일에 추가해야 합니다. 세부 정보 부록 **F.** ***AAP Operator*****에 관리되는 추가 네임스페이스 추가**.

참고

Ansible Automation Platform Operator의 기본 리소스 값은 일반적인 설치에 적합합니다. 그러나 많은 자동화 컨트롤러 및 자동화 허브 환경을 배포하는 경우 **subscription.spec.config.resources** 를 사용하여 서브스크립션 사양 내에서 **Ansible Automation Platform Operator**의 리소스 임계값을 늘리는 것이 좋습니다. 이렇게 하면 **Operator**에 증가된 워크로드를 처리하고 성능 문제를 방지할 수 있는 충분한 리소스가 있습니다.

6장. 자동화 컨트롤러 설치

Ansible Automation Platform Operator 설치가 완료되면 다음 단계는 **Red Hat OpenShift** 클러스터 내에 자동화 컨트롤러를 설치합니다.



참고

리소스 요청 및 제한 값은 이 참조 환경에 따라 다릅니다. **Red Hat OpenShift** 환경의 값을 올바르게 계산하려면 **3장.** *시작하기 전* 섹션을 읽으십시오.

주의

자동화 컨트롤러 인스턴스가 제거되면 연결된 **PVC**(영구 볼륨 클레임)가 자동으로 삭제되지 않습니다. 이로 인해 새 배포의 이름이 이전 배포와 동일한 경우 마이그레이션 중 문제가 발생할 수 있습니다. 동일한 네임스페이스에 새 자동화 컨트롤러 인스턴스를 배포하기 전에 이전 **PVC**를 제거하는 것이 좋습니다. 이전 배포 **PVC**를 제거하는 단계는 **부록 B.** *이전 **AAP** 설치에서 기존 **PVC** 삭제* 에서 확인할 수 있습니다.

- 클러스터 인증 정보를 사용하여 **Red Hat OpenShift** 웹 콘솔에 로그인합니다.
- 왼쪽 탐색 메뉴에서 **Operator** → 설치된 **Operator** 를 선택하고 **Ansible Automation Platform** 을 선택합니다.
- 자동화 컨트롤러 탭으로 이동한 다음 자동화 컨트롤러 생성 을 클릭합니다.
- 양식 보기 내에서 이름 (예: ***my-automation-controller***)을 제공하고 고급 구성 을 선택하여 추가 옵션을 확장합니다.
- 추가 구성 내에서 시작하기 전 섹션에서 계산된 각 컨테이너에 대한 적절한 리소스 요구 사항을 설정합니다.
  - 웹 컨테이너 리소스 요구 사항확장

- 제한: CPU 코어: 2000m, 메모리: 1.5Gi
- 요청: CPU 코어: 500m, 메모리: 1.5Gi

- Task 컨테이너 리소스 요구 사항확장
  - 제한: CPU 코어: 4000m, 메모리: 8Gi
  - 요청: CPU 코어: 1000m, 메모리: 8Gi

- EE 컨트롤 플레인 컨테이너 리소스 요구 사항확장
  - 제한: CPU 코어: 500m, 메모리: 400Mi
  - 요청: CPU 코어: 100m, 메모리: 400Mi

- Redis 컨테이너 리소스 요구 사항을확장
  - 제한: CPU 코어: 500m, 메모리: 1.5Gi
  - 요청: CPU 코어: 250m, Memory: 1.5Gi

- PostgreSQL 컨테이너 리소스 요구 사항확장
  - 제한: CPU 코어: 1000m, 메모리: 1Gi
  - 요청: CPU 코어: 500m, 메모리: 1Gi

- **Create AutomationController** 페이지 상단에서 **YAML** 보기를전환합니다.
  - **spec:** 섹션 내에서 **extra_settings** 매개변수를 추가하여 **3장.** *시작하기 전* 섹션에 계산된 **AWX_CONTROL_NODE_TASK_IMPACT** 값을 전달합니다.

```
spec:
...
  extra_settings:
    - setting: AWX_CONTROL_NODE_TASK_IMPACT
      value: "5"
```

- **YAML** 보기에서 **spec** 섹션에 다음을 추가하여 제어 **Pod**의 전용 노드를 추가합니다.

```
spec:
...
  node_selector: |
    aap_node_type: control
  topology_spread_constraints: |
    - maxSkew: 1
      topologyKey: "kubernetes.io/hostname"
      whenUnsatisfiable: "ScheduleAnyway"
      labelSelector:
        matchLabels:
          aap_node_type: control
  tolerations: |
    - key: "dedicated"
      operator: "Equal"
      value: "AutomationController"
      effect: "NoSchedule"
```

참고

제어 **Pod**를 실행해야 하는 적절한 전용 작업자 노드에 대한 노드 레이블 및 테인트가 있어야 합니다. 설정할 세부 정보는 부록 ***C. Red Hat OpenShift 노드에 레이블 및 테인트 적용*** 에서 확인할 수 있습니다.

- 생성 버튼을 클릭합니다.

7장. 자동화 허브 설치

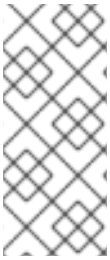
Ansible Automation Platform Operator 설치가 완료되면 다음 단계에서 **Red Hat OpenShift** 클러스터 내에서 자동화 허브를 설치합니다.

참고

리소스 요청 및 제한 값은 이 참조 환경에 따라 다릅니다. **Red Hat OpenShift** 환경의 값을 올바르게 계산하려면 **3장.** *시작하기 전* 섹션을 읽으십시오.

주의

자동화 허브 인스턴스를 제거하면 연결된 **PVC**(영구 볼륨 클레임)가 자동으로 삭제되지 않습니다. 이로 인해 새 배포의 이름이 이전 배포와 동일한 경우 마이그레이션 중 문제가 발생할 수 있습니다. 동일한 네임스페이스에 새 자동화 허브 인스턴스를 배포하기 전에 이전 **PVC**를 제거하는 것이 좋습니다. 이전 배포 **PVC**를 제거하는 단계는 부록 **B.** *이전 **AAP** 설치에서 기존 **PVC** 삭제* 에서 확인할 수 있습니다.



참고

Automation Hub에는 여러 **Pod**가 컬렉션과 같은 공유 콘텐츠에 액세스할 수 있도록 작업을 위해 **ReadWriteMany** 파일 기반 스토리지, **Azure Blob** 스토리지 또는 **Amazon S3** 호환 스토리지가 필요합니다.

- 클러스터 인증 정보를 사용하여 **Red Hat OpenShift** 웹 콘솔에 로그인합니다.
- 왼쪽 탐색 메뉴에서 **Operator** → 설치된 **Operator** 를 선택하고 **Ansible Automation Platform** 을 선택합니다.
- **Automation Hub** 탭으로 이동한 다음 **Create AutomationHub** 를 클릭합니다.
- 양식 보기에서

- 이름을 지정합니다 (예: *my-automation-hub*)

- 스토리지 유형에서 **ReadWriteMany** 호환 스토리지를 선택합니다.

참고

이 참조 환경에서는 **Amazon S3**를 **ReadWriteMany** 스토리지로 사용합니다. **Amazon S3** 버킷을 생성하는 방법에 대한 자세한 내용은 부록 D. *Amazon S3 버킷 생성* 에서 확인할 수 있습니다.

  - **S3** 스토리지 시크릿을 제공합니다. 부록 E. *AWS S3 시크릿 생성* 내에서 생성하는 방법에 대한 세부 정보입니다.

- 고급 구성 을 선택하여 추가 옵션을 확장합니다.

- **PostgreSQL** 컨테이너 스토리지 요구 사항(관리 인스턴스 사용 시)

  - 스토리지 제한을 **50Gi**로 설정

  - 스토리지 요청을 **8Gi**로 설정

- **PostgreSQL** 컨테이너 리소스 요구 사항(관리 인스턴스 사용 시)

  - 제한: **CPU** 코어: **500m**, 메모리: **1Gi**

  - 요청: **CPU** 코어: **200m**, **Memory**: **1Gi**

- **Redis** 배포 구성 내에서 고급 구성 을 선택합니다.

  - 메모리 내 데이터 저장소 리소스 요구 사항선택

- 제한: CPU 코어: 250m, 메모리: 200Mi
- 요청: CPU 코어: 100m, 메모리: 200Mi

- API 서버 구성 내에서 고급 구성 을 선택합니다.
  - API 서버 리소스 요구 사항선택
    - 제한: CPU 코어: 250m, 메모리: 400Mi
    - 요청: CPU 코어: 150m, 메모리: 400Mi
- 콘텐츠 서버 구성 내에서 고급 구성 을 선택합니다.
  - 콘텐츠 서버 리소스 요구 사항선택
    - 제한: CPU 코어: 250m, 메모리: 400Mi
    - 요청: CPU 코어: 100m, 메모리: 400Mi
- Worker 구성 내에서 고급 구성을 선택합니다.
  - Worker 리소스 요구 사항선택
    - 제한: CPU 코어: 1000m, 메모리: 3Gi
    - 요청: CPU 코어: 500m, Memory: 3Gi

- **생성 버튼을 클릭합니다.**

8장. 자동화 컨트롤러 대시보드에 로그인

자동화 컨트롤러를 성공적으로 설치하면 다음 단계를 통해 대시보드에 액세스할 수 있습니다.

- Red Hat OpenShift 웹 콘솔의 왼쪽 탐색 메뉴에서 **Operators**를 선택합니다.
- **Ansible Automation Platform** 을 선택합니다.
- **Operator** 세부 정보에서 자동화 컨트롤러 탭을 선택합니다.
- 설치된 자동화 컨트롤러의 이름을 선택합니다.
- 자동화 컨트롤러 개요 내에서 **URL, Admin** 사용자 및 **Admin** 암호를 포함한 세부 정보가 제공됩니다.

9장. 자동화 허브 대시보드에 로그인

- **Red Hat OpenShift** 웹 콘솔의 왼쪽 탐색 메뉴에서 **Operators**를 선택합니다.
- **Ansible Automation Platform** 을 선택합니다.
- **Operator** 세부 정보에서 **Automation Hub** 탭을 선택합니다.
- 설치된 자동화 허브의 이름을 선택합니다.
- **Automation Hub** 개요 내에서 **URL, Admin** 사용자 및 **Admin** 암호를 포함한 세부 정보가 제공됩니다.

10장. ANSIBLE AUTOMATION PLATFORM 모니터링

Ansible Automation Platform을 성공적으로 설치한 후에는 상태 유지 관리와 주요 메트릭을 모니터링하는 기능을 우선시하는 것이 중요합니다.

이 섹션에서는 **Red Hat OpenShift**에 상주하는 새로 설치된 **Ansible Automation Platform** 환경에서 제공하는 **API** 메트릭을 모니터링하는 방법을 중점적으로 설명합니다.

10.1. API 메트릭을 모니터링하는 데 사용되는 것은 무엇입니까?

Prometheus 및 **Grafana.**

Prometheus는 메트릭을 수집하고 집계하기 위한 오픈 소스 모니터링 솔루션입니다. 파트너 **Prometheus**의 모니터링 기능은 데이터 분석을 실행하고 사용자 지정 가능한 대시보드에서 메트릭을 가져오는 데 필요한 오픈 소스 솔루션인 **Grafana**를 사용하여 **Ansible Automation Platform**의 상태 및 상태를 추적하기 위해 메트릭을 실시간으로 시각화할 수 있습니다.

10.2. 어떤 메트릭을 볼 수 있습니까?

Grafana 사전 빌드 대시보드가 표시됩니다.

- **Ansible Automation Platform** 버전
- 컨트롤러 노드 수
- 라이센스에서 사용 가능한 호스트 수
- 사용되는 호스트 수
- 총 사용자
- 작업 성공

- 작업 실패
- 작업 실행 유형별 수량
- 실행 중인 작업 수 및 보류 중인 작업 수를 보여주는 그래픽
- 워크플로우, 호스트, 인벤토리, 작업, 프로젝트, 조직 등의 양을 보여주는 툴의 증가를 보여주는 그래프입니다.

이 Grafana 대시보드는 관심 있는 다른 메트릭을 캡처하도록 사용자 지정할 수 있습니다. 그러나 Grafana 대시보드 사용자 지정은 이 참조 아키텍처에서 다루지 않습니다.

10.3. ANSIBLE 플레이북을 통한 설치

사용자 지정 Grafana 대시보드를 사용하여 Prometheus를 통해 Ansible Automation Platform을 모니터링하는 프로세스는 몇 분 내에 설치할 수 있습니다. 다음은 사전 빌드된 Ansible 플레이북을 활용하여 이를 수행하는 단계를 제공합니다.

Ansible 플레이북을 성공적으로 실행하려면 다음 단계가 필요합니다.

- 자동화 컨트롤러 내에서 사용자 정의 인증 정보 유형 생성
- 자동화 컨트롤러 내에 kubeconfig 인증 정보 생성
- Ansible 플레이북을 실행하기 위한 프로젝트 및 작업 템플릿 생성

10.3.1. 사용자 정의 인증 정보 유형 생성

Ansible Automation Platform 대시보드 내에서

1. Administration(관리)Credential Types(인증 정보 유형)에서 blue Add 버튼을 클릭합니다.

2. 이름 지정(예: ***Kubeconfig***)

3. 입력 구성에서 다음 **YAML**을 입력합니다.

```
fields:
  - id: kube_config
    type: string
    label: kubeconfig
    secret: true
    multiline: true
```

4. 인젝터 구성에서 다음 **YAML**을 입력합니다.

```
env:
  K8S_AUTH_KUBECONFIG: '{{ tower.filename.kubeconfig }}'
file:
  template.kubeconfig: '{{ kube_config }}'
```

5. 저장을 클릭합니다.

10.3.2. kubeconfig 인증 정보 생성

Ansible Automation Platform 대시보드 내에서

1. **ResourcesCredentials** 에서 **blue Add** 버튼을 클릭합니다.

2. 이름을 입력합니다 (예: ***OpenShift-Kubeconfig***)

3. 인증 정보 유형 드롭다운에서 **Kubeconfig** 를 선택합니다.

4. 유형 세부 정보 텍스트 상자에 **Red Hat OpenShift** 클러스터에 대한 **kubeconfig** 파일을 삽입합니다.

5. 저장을 클릭합니다.

10.3.3. 프로젝트 생성

Ansible Automation Platform 대시보드 내에서

1. **Resources CryostatProjects** 에서 파란색 추가 버튼을 클릭합니다.
2. 이름 제공(예: *모니터링 **AAP** 프로젝트)*
3. **Default** 를 조직으로 선택합니다.
4. 실행 환경으로 기본 실행 환경을 선택합니다.
5. 소스 제어 인증 정보 유형으로 **Git** 을 선택합니다.
6. 유형 상세 정보 에서,
   a. 소스 제어 **URL** 추가 (**https://github.com/ansible/aap_ocp_refarch**)
7. 옵션 내에서,
   a. 시작 시 정리, 삭제, 버전 업데이트를 선택합니다.
8. 저장을 클릭합니다.

10.3.4. 작업 템플릿 생성 및 Ansible Playbook 실행

Ansible Automation Platform dsahboard에서

1. **ResourcesTemplates** 에서 파란색 추가 추가 작업 템플릿을클릭합니다.

2. 이름 제공(예: ***Monitoring AAP Job***)

3. 작업 유형으로 실행을 선택합니다.

4. 인벤토리로 데모 인벤토리 를 선택합니다.

5. **Monitoring AAP Project as the Project** 를 선택합니다.

6. 실행 환경으로 기본 실행 환경을 선택합니다.

7. 플레이북으로 ***aap-prometheus-grafana/playbook.yml*** 을 선택합니다.

8. 인증 정보를 선택하고 머신 의 카테고리를 **Kubeconfig** 로 전환합니다.

9. **Red Hat OpenShift** 클러스터에 액세스하기 위한 적절한 ***kubeconfig*** (예: ***OpenShift-Kubeconfig***)를 선택합니다.

10. 선택적 단계: 변수 내에서 다음 변수 를 수정할 수 있습니다.

    a. **prometheus_namespace: *<your-specified-value>***

    b. **ansible_namespace: *<your-specified-value>***

11. 저장을 클릭합니다.

12. 시작을 클릭하여 **Ansible** 플레이북을 실행합니다.

13. **Grafana** 및 **Prometheus**에 로그인하기 위한 세부 정보가 작업 출력에 표시됩니다.

부록 A. 작성자 정보

Roger Lopez

Roger Lopez is a Principal Technical Marketing Manager bringing 10+ years of computer industry experience delivering high-value solutions used by our sales, marketing and engineering teams to develop best practice documentation & methods for internal and external customers. He is a Red Hat Certified Engineer (RHCE) with experience building solutions around Ansible, OpenShift and OpenStack.

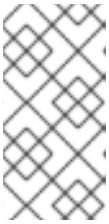
부록 B. 이전 AAP 설치에서 기존 PVC 삭제

1. 터미널 창을 열고 **Red Hat OpenShift** 클러스터에 로그인합니다(예: **KUBECONFIG** 파일 내보내기).

```
$ export KUBECONFIG=/path/to/kubeconfig
```

2. 지정된 **Ansible Automation Platform** 네임스페이스의 **PVC** 목록(예: **aap**) 확인

```
$ oc get pvc -n aap
```



참고

그러면 이름, 상태, 용량, 액세스 모드 및 스토리지 클래스를 포함하여 **aap** 네임스페이스 내의 모든 **PVC** 목록이 표시됩니다.

3. 목록에서 제거하려는 **PVC** 확인

4. **oc delete** 명령을 사용하여 **PVC** 제거

```
oc delete pvc <pvc-name-to-remove> -n aap
```

aap 네임스페이스에서 사용 가능한 **PVC** 목록을 확인하고 **PVC**가 더 이상 표시되지 않는지 확인합니다.

```
$ oc get pvc -n aap
```

부록 C. RED HAT OPENSHIFT 노드에 레이블 및 테인트 적용

제어 **Pod**가 전용 **Red Hat OpenShift** 노드에서 실행되도록 하려면 적절한 라벨과 테인트를 지정된 노드로 설정해야 합니다.

이 예제에서는 역할 **worker** 가 **aap_node_type=control** 레이블을 사용하는 **Red Hat OpenShift** 노드 중 하나를 선택합니다.

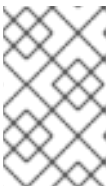
1. 실행 중인 노드에 레이블을 지정할 노드 중 하나의 이름을 가져옵니다.

```
$ oc get nodes
```

2. 목록에서 노드를 선택하고 해당 이름(예: **worker1**)을 기록해 둡니다.

3. **aap_node_type=control** 라벨을 노드에 적용

```
$ oc label node <node-name> aap_node_type=control
```



참고

& lt;node-name >을 레이블을 지정할 노드 이름으로 바꿉니다.

4. 다음과 같이 라벨 생성을 확인합니다.

```
$ oc get nodes --show-labels | grep <node-name>
```

레이블이 생성되면 다음 단계는 이미 레이블을 생성한 작업자 노드에 **NoSchedule** 테인트를 추가하는 것입니다.

다음 명령은 **NoSchedule** 테인트를 노드에 추가합니다.

```
oc adm taint nodes <node-name> dedicated=AutomationController:NoSchedule
```

dedicated: 테인트를 식별하는 임의의 문자열인 테인트의 키입니다.

AutomationController: 이 값은 테인트에 지정된 임의의 값입니다.

NoSchedule: 이 테인트를 허용하지 않는 **Pod**를 지정하지 않는 테인트의 영향으로 이 테인트는 이 노드에 예약됩니다.

이 테인트를 노드에 적용하여 **Kubernetes** 스케줄러에 테인트를 허용하는 특정 유형의 워크로드에 대해 이 노드를 예약하도록 지시합니다. 이 경우 **dedicated=AutomationController** 허용 오차를 사용하여 워크로드에 노드를 예약합니다.

5. 테인트가 적용되었는지 확인합니다.

```
$ oc get nodes \
-o jsonpath='{range.items[*]}{@.metadata.name}{"\t"}{@.spec.taints[*].key}:
{@.spec.taints[*].value}{"\n"}{end}' \
| grep AutomationController
```

부록 D. AMAZON S3 버킷 생성

1. 터미널을 열고 **AWS CLI**가 **AWS** 인증 정보로 설치 및 구성되었는지 확인합니다.

2. 다음 명령을 실행하여 새 **S3** 버킷을 생성합니다.

```
$ aws s3 mb s3://<bucket-name> --region <region-name>
```

주의

버킷 이름은 고유한 이름이어야 합니다.

3. 다음 명령을 실행하여 버킷이 성공적으로 생성되었는지 확인합니다.

```
$ aws s3 ls | grep <bucket-name>
```

부록 E. AWS S3 시크릿 생성

Red Hat OpenShift에서 시크릿을 저장하고 **Amazon S3**와 같은 외부 서비스로 인증하는 데 사용할 수 있는 기능이 있습니다. 이 참조 환경은 자동화 허브를 실행하는 데 필요한 **ReadWriteMany** 요구 사항을 충족하기 위해 **Amazon S3** 버킷을 활용합니다.

다음 단계에서는 *7장. 자동화 허브 설치* 장에서 사용할 **Red Hat OpenShift** 클러스터 내에 시크릿을 생성하는 방법을 보여줍니다.

```
$ cat s3-secret.yml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: s3-secret
  namespace: aap
stringData:
  s3-access-key-id: my_key
  s3-secret-access-key: my_access_key
  s3-bucket-name: my_bucket
  s3-region: my_region
```

```
$ oc create -f s3-secret.yml
```

부록 F. AAP OPERATOR에 관리되는 추가 네임스페이스 추가

다음은 **aap** 네임스페이스에 있는 기존 **AAP Operator**에 추가 관리 네임스페이스를 추가하는 단계입니다.

- **Red Hat OpenShift UI**에 로그인합니다.
- **Operator**에서**Installed Operator** 를 선택하고 **Ansible Automation Platform**을 선택합니다.
- 세부 정보 페이지 내에서 **ClusterServiceVersion Details** 까지 아래로 스크롤
  - 오른쪽 열에서 **OperatorGroup**을 클릭합니다.
- **OperatorGroup** 세부 정보 내에서 **YAML**을 선택합니다.
  - **spec** 섹션에서 추가 네임스페이스(예: **aap-devel**)를 추가합니다.

```
spec:
...
  targetNamespaces:
    - aap
    - aap-devel
```

- 저장을클릭합니다.

참고

OperatorGroup 사양 파일에 추가하기 전에 대상 네임스페이스가 이미 있어야 합니다.

부록 G. 참고 자료

- 모든 사용자가 Kubernetes 메모리 제한, OOMKilled Pod 및 internal 당사자에 대해 알아야 할 사항
- Pulp 프로젝트 하드웨어 요구 사항
- Operator 기반 설치에 대한 Red Hat Ansible Automation Platform 성능 고려 사항
- OpenShift Container Platform에 Red Hat Ansible Automation Platform Operator 배포
- Pulp Operator 스토리지 구성
- AWX Operator
- 유휴 PostgreSQL 연결에서 사용하는 리소스
- Operator scopeoping with OperatorGroups
- Ansible Tower 및 Grafana 대시보드

부록 H. REVISION HISTORY

고침 1.0-0	2023-03-07	Roger Lopez

- Initial Release