OpenShift Container Platform 4.8

인증 및 권한 부여

사용자 및 서비스에 대한 사용자 인증 및 액세스 제어 구성

Last Updated: 2023-06-06

OpenShift Container Platform 4.8 인증 및 권한 부여

사용자 및 서비스에 대한 사용자 인증 및 액세스 제어 구성

법적 공지

초록

이 문서에서는 OpenShift Container Platform에서 ID 공급자를 정의하는 방법을 설명합니다. 또한 클러스터를 보호하기 위해 역할 기반 액세스 제어를 구성하는 방법도 알아봅니다.

차례

1장. 인증 및 권한 부여 개요

1.1. OPENSHIFT CONTAINER PLATFORM 인증 및 권한 부여에 대한 일반 용어

이 용어집은 OpenShift Container Platform 인증 및 권한 부여에 사용되는 일반적인 용어를 정의합니다.

인증

인증에 따라 OpenShift Container Platform 클러스터에 대한 액세스 권한이 결정되며 인증된 사용자만 OpenShift Container Platform 클러스터에 액세스할 수 있습니다.

권한 부여

권한 부여는 식별된 사용자에게 요청된 작업을 수행할 수 있는 권한이 있는지 여부를 결정합니다.

전달자 토큰

전달자 토큰은 헤더 **Authorization을 사용하여 API에 인증하는 데 사용됩니다. 전달자 <token>**.

Cloud Credential Operator

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 CRD(사용자 지정 리소스 정의)로 관리합니다.

구성 맵

구성 맵에서는 구성 데이터를 Pod에 삽입하는 방법을 제공합니다. 구성 맵에 저장된 데이터를 **ConfigMap** 유형의 볼륨에서 참조할 수 있습니다. Pod에서 실행되는 애플리케이션에서는 이 데이터를 사용할 수 있습니다.

컨테이너

소프트웨어 및 모든 종속 항목을 구성하는 경량 및 실행 가능한 이미지입니다. 컨테이너는 운영 체제를 가상화하므로 데이터 센터, 퍼블릭 또는 프라이빗 클라우드 또는 로컬 호스트에서 컨테이너를 실행할 수 있습니다.

CR(사용자 정의 리소스)

CR은 Kubernetes API의 확장입니다.

group

그룹은 사용자 집합입니다. 그룹은 여러 사용자에게 한 번 권한을 부여하는 데 유용합니다.

HTPasswd

htpasswd는 HTTP 사용자 인증을 위해 사용자 이름 및 암호를 저장하는 파일을 업데이트합니다.

Keystone

Keystone은 ID, 토큰, 카탈로그 및 정책 서비스를 제공하는 RHOSP(Red Hat OpenStack Platform) 프로젝트입니다.

LDAP(Lightweight Directory Access Protocol)

LDAP는 사용자 정보를 쿼리하는 프로토콜입니다.

수동 모드

수동 모드에서 사용자는 CCO(Cloud Credential Operator) 대신 클라우드 인증 정보를 관리합니다.

Mint 모드

Mint 모드는 지원되는 플랫폼에서 사용할 CCO(Cloud Credential Operator)에 대한 기본 권장 모범 사례 설정입니다. 이 모드에서 CCO는 제공된 관리자 수준 클라우드 인증 정보를 사용하여 필요한 특정 권한만으로 클러스터의 구성 요소에 대한 새 인증 정보를 생성합니다.

namespace

네임스페이스는 모든 프로세스에 표시되는 특정 시스템 리소스를 격리합니다. 네임스페이스 내에서 해당 네임스페이스의 멤버인 프로세스만 해당 리소스를 볼 수 있습니다.

node

노드는 OpenShift Container Platform 클러스터의 작업자 시스템입니다. 노드는 VM(가상 머신) 또는 물리적 머신입니다.

OAuth 클라이언트

OAuth 클라이언트는 전달자 토큰을 얻는 데 사용됩니다.

OAuth 서버

OpenShift Container Platform 컨트롤 플레인에는 구성된 ID 공급자의 사용자 ID를 결정하고 액세스 토큰을 생성하는 기본 제공 OAuth 서버가 포함되어 있습니다.

OpenID Connect

OpenID Connect는 사용자가 SSO(Single Sign-On)를 사용하여 OpenID 공급자를 사용하는 사이트에 액세스하도록 사용자를 인증하는 프로토콜입니다.

Passthrough 모드

Passthrough 모드에서 CCO(Cloud Credential Operator)는 제공된 클라우드 인증 정보를 클라우드 인증 정보를 요청하는 구성 요소에 전달합니다.

Pod

Pod는 Kubernetes에서 가장 작은 논리 단위입니다. Pod는 작업자 노드에서 실행할 하나 이상의 컨테이너로 구성됩니다.

Regular users

처음 로그인하거나 API를 통해 클러스터에서 자동으로 생성되는 사용자입니다.

요청 헤더

요청 헤더는 HTTP 요청 컨텍스트에 대한 정보를 제공하는 데 사용되는 HTTP 헤더이므로 서버가 요청의 응답을 추적할 수 있습니다.

RBAC(역할 기반 액세스 제어)

클러스터 사용자와 워크로드가 역할을 실행하는 데 필요한 리소스에만 액세스할 수 있도록 하는 주요 보안 제어입니다.

서비스 계정

서비스 계정은 클러스터 구성 요소 또는 애플리케이션에서 사용합니다.

System users

클러스터가 설치될 때 자동으로 생성되는 사용자입니다.

사용자

사용자는 API에 요청할 수 있는 엔티티입니다.

1.2. OPENSHIFT CONTAINER PLATFORM의 인증 정보

OpenShift Container Platform 클러스터에 대한 액세스를 제어하기 위해 클러스터 관리자는 사용자 인증을 구성하고 승인된 사용자만 클러스터에 액세스할 수 있도록 할 수 있습니다.

OpenShift Container Platform 클러스터와 상호 작용하려면 먼저 OpenShift Container Platform API에 대해 어떤 방식으로든 인증해야 합니다. OpenShift Container Platform API에 대한 요청에 OAuth 액세스 토큰 또는 X.509 클라이언트 인증서를 제공하여 인증할 수 있습니다.

참고

유효한 액세스 토큰 또는 인증서가 표시되지 않으면 요청이 인증되지 않고 HTTP 401 오류가 표시됩니다.

관리자는 다음 작업을 통해 인증을 구성할 수 있습니다.

- ID 공급자 구성: OpenShift Container Platform에서 지원되는 ID 공급자를 정의하고 클러스터에 추가할 수 있습니다.
- 내부 OAuth 서버 구성: OpenShift Container Platform 컨트롤 플레인에는 구성된 ID 공급자의 사용자 ID를 결정하고 액세스 토큰을 생성하는 기본 제공 OAuth 서버가 포함되어 있습니다. 토큰 기간 및 비활성 타임아웃을 구성할 수 있습니다.

참고

사용자는 사용자가 소유한 OAuth 토큰을 보고 관리할 수 있습니다.

- OAuth 클라이언트 등록: OpenShift Container Platform에는 몇 가지 기본 OAuth 클라이언트 가 포함되어 있습니다. 추가 OAuth 클라이언트를 등록하고 구성할 수 있습니다.

참고

사용자가 OAuth 토큰에 대한 요청을 보내면 토큰을 수신하고 사용하는 기본 또는 사용자 지정 OAuth 클라이언트를 지정해야 합니다.

- Cloud Credentials Operator 를 사용하여 클라우드 공급자 인증 정보 관리: 클러스터 구성 요소에서는 클라우드 공급자 인증 정보를 사용하여 클러스터 관련 작업을 수행하는 데 필요한 권한을 얻습니다.
- 시스템 관리자 가장: 시스템 admin 사용자를 가장 하여 클러스터 관리자 권한을 사용자에게 부여할 수 있습니다.

1.3. OPENSHIFT CONTAINER PLATFORM의 인증 정보

권한 부여는 식별된 사용자가 요청한 작업을 수행할 수 있는 권한이 있는지 여부를 결정합니다.

관리자는 규칙, 역할, 바인딩과 같은 RBAC 개체를 사용하여 권한을 정의하고 사용자에게 할당할 수 있습니다. OpenShift Container Platform에서 인증이 작동하는 방식을 이해하려면 권한 부여 비우기를 참조하십시오.

프로젝트 및 네임스페이스 를 통해 OpenShift Container Platform 클러스터에 대한 액세스를 제어할 수도 있습니다.

클러스터에 대한 사용자 액세스 제어와 함께 포드에서 수행할 수 있는 작업과 SCC(보안 컨텍스트 제약 조건) 를 사용하여 액세스할 수 있는 리소스를 제어할 수 있습니다.

다음 작업을 통해 OpenShift Container Platform에 대한 권한을 관리할 수 있습니다.

- 로컬 및 클러스터 역할 및 바인딩 보기.
- 로컬 역할을 생성하여 사용자 또는 그룹에 할당합니다.
- 클러스터 역할을 생성하고 사용자 또는 그룹에 할당합니다. OpenShift Container Platform에는 기본 클러스터 역할 세트가 포함되어 있습니다. 추가 클러스터 역할을 생성하여 사용자 또는 그룹에 추가할 수 있습니다.
- cluster-admin 사용자를 생성합니다. 기본적으로 클러스터에는 **kubeadmin** 이라는 클러스터 관리자가 하나만 있습니다. 다른 클러스터 관리자를 생성할 수 있습니다. 클러스터 관리자를 생성하기 전에 ID 공급자를 구성해야 합니다.

참고

클러스터 admin 사용자를 생성한 후 기존 kubeadmin 사용자를 삭제 하여 클러스터 보안을 개선합니다.

- 서비스 계정 생성: 서비스 계정을 사용하면 일반 사용자의 자격 증명을 공유하지 않고도 API 액세스를 유연하게 제어할 수 있습니다. 사용자는 애플리케이션에서 서비스 계정을 생성 및 사용할 수 있으며 OAuth 클라이언트로 도 사용할 수 있습니다.
- 범위 지정 토큰: 범위가 지정된 토큰은 특정 작업만 수행할 수 있는 특정 사용자로 식별되는 토큰입니다. 범위가 지정된 토큰을 생성하여 일부 권한을 다른 사용자 또는 서비스 계정에 위임할 수 있습니다.
- LDAP 그룹 동기화: LDAP 서버에 저장된 그룹을 OpenShift Container Platform 사용자 그룹과 동기화 하여 한 곳에서 사용자 그룹을 관리할 수 있습니다.

2장. 인증 이해

OpenShift Container Platform과 상호 작용하려는 사용자는 먼저 클러스터에 인증해야 합니다. 그러면 인증 계층에서 OpenShift Container Platform API에 대한 요청과 관련된 사용자를 확인합니다. 그런 다음 권한 부여 계층에서 요청한 사용자에 대한 정보로 요청의 허용 여부를 결정합니다.

관리자는 OpenShift Container Platform에 대한 인증을 구성할 수 있습니다.

2.1. 사용자

OpenShift Container Platform에서 *사용자*는 OpenShift Container Platform API에 요청할 수 있는 엔티티입니다. OpenShift Container Platform **User** 오브젝트는 행위자를 나타냅니다. 이 행위자에는 시스템에서 행위자 또는 행위자 그룹에 역할을 추가하여 권한을 부여할 수 있습니다. 일반적으로 OpenShift Container Platform과 상호 작용하는 개발자 또는 관리자 계정을 의미합니다.

다음과 같이 여러 유형의 사용자가 존재할 수 있습니다.

사용자 유형	설명
Regular users	상호 작용을 가장 많이 하는 OpenShift Container Platform 사용자입니다. 일반 사용자는 처음 로그인할 때 시스템에서 자동으로 생성되며 API를 통해 생성할 수도 있습니다. 일반 사용자는 **User** 오브젝트로 표시됩니다. 예:**joe alice**
System users	대부분의 시스템 사용자는 주로 인프라와 API 간의 안전한 상호 작용을 목적으로 인프라가 정의될 때 자동으로 생성됩니다. 여기에는 클러스터 관리자(전체 액세스 권한 보유), 노드별 사용자, 라우터 및 레지스트리용 사용자를 비롯하여 기타 다양한 사용자가 포함됩니다. 마지막으로, 인증되지 않은 요청에 기본적으로 사용되는 **anonymous** 시스템 사용자가 있습니다. 예: **system:admin system:openshift-registry system:node:node1.example.com**
Service accounts	프로젝트와 관련된 특수한 시스템 사용자입니다. 일부는 프로젝트가 처음 생성될 때 자동으로 생성되지만 프로젝트 관리자가 각 프로젝트 콘텐츠에 대한 액세스 권한을 정의하기 위해 추가로 생성할 수도 있습니다. 서비스 계정은 **ServiceAccount** 오브젝트로 표시됩니다. 예: **system:serviceaccount:default:deployer system:serviceaccount:foo:builder**

각 사용자가 OpenShift Container Platform에 액세스하려면 어떠한 방식으로든 인증해야 합니다. 인증되지 않았거나 인증이 유효하지 않은 API 요청은 **anonymous** 시스템 사용자의 요청으로 인증됩니다. 인증 후 정책에 따라 사용자가 수행할 수 있는 작업이 결정됩니다.

2.2. 그룹

사용자는 하나 이상의 *그룹*에 할당될 수 있으며, 각 그룹은 특정 사용자 집합을 나타냅니다. 그룹은 권한 부여 정책을 관리하여 여러 사용자에게 한꺼번에 권한을 부여할 때 유용합니다(예: 사용자에게 개별적으로 오브젝트 액세스 권한을 부여하는 대신 특정 프로젝트 내의 여러 오브젝트에 대한 액세스 허용).

명시적으로 정의된 그룹 외에도 클러스터에서 자동으로 프로비저닝하는 시스템 그룹 또는 *가상 그룹*이 있습니다.

다음과 같은 기본 가상 그룹이 가장 중요합니다.

가상 그룹	설명
system:authenticated	인증된 모든 사용자와 자동으로 연결됩니다.
system:authenticated:oauth	OAuth 액세스 토큰을 사용하여 인증된 모든 사용자와 자동으로 연결됩니다.
system:unauthenticated	인증되지 않은 모든 사용자와 자동으로 연결됩니다.

2.3. API 인증

OpenShift Container Platform API에 대한 요청은 다음과 같은 방법으로 인증됩니다.

OAuth 액세스 토큰

- ***<namespace_route>*/oauth/authorize** 및 ***<namespace_route>*/oauth/token** 끝점을 사용하여 OpenShift Container Platform OAuth 서버에서 가져옵니다.
- **인증: 전달자...** 헤더.
- WebSocket 요청의 경우 **base64url.bearer.authorization.k8s.io.<base64url-encoded-token>** 형식의 WebSocket 하위 프로토콜 헤더로 전송됩니다.

X.509 클라이언트 인증서

- API 서버에 대한 HTTPS 연결이 필요합니다.
- 신뢰할 수 있는 인증 기관 번들과 대조하여 API 서버에서 확인합니다.
- API 서버는 인증서를 작성하고 컨트롤러에 분배하여 자체적으로 인증합니다.

유효하지 않은 액세스 토큰 또는 유효하지 않은 인증서가 있는 요청은 **401** 오류와 함께 인증 계층에서 거부됩니다.

액세스 토큰이나 인증서가 없는 경우 인증 계층은 **system:anonymous** 가상 사용자 및 **system:unauthenticated** 가상 그룹을 요청에 할당합니다. 그러면 권한 부여 계층에서 익명 사용자가 할 수 있는 요청(있는 경우)을 결정합니다.

2.3.1. OpenShift Container Platform OAuth 서버

OpenShift Container Platform 마스터에는 내장 OAuth 서버가 포함되어 있습니다. 사용자는 API 인증을 위해 OAuth 액세스 토큰을 가져옵니다.

사용자가 새 OAuth 토큰을 요청하면 OAuth 서버는 구성된 ID 공급자를 사용하여 요청한 사람의 ID를 확인합니다.

그런 다음 해당 ID와 매핑되는 사용자를 결정하고 그 사용자를 위한 액세스 토큰을 만들어 제공합니다.

2.3.1.1. OAuth 토큰 요청

OAuth 토큰을 요청할 때마다 토큰을 받고 사용할 OAuth 클라이언트를 지정해야 합니다. OpenShift Container Platform API를 시작하면 다음 OAuth 클라이언트가 자동으로 생성됩니다.

OAuth 클라이언트	사용법
openshift-browser-client	대화형 로그인을 처리할 수 있는 사용자 에이전트를 사용하여 **<namespace_route>/oauth/token/request** 에서 토큰을 요청합니다. [1]
openshift-challenging-client	**WWW-Authenticate** 챌린지를 처리할 수 있는 사용자 에이전트로 토큰을 요청합니다.

1. **<namespace_route>**는 네임스페이스 경로를 나타냅니다. 다음 명령을 실행하여 확인할 수 있습니다.

```
$ oc get route oauth-openshift -n openshift-authentication -o json | jq .spec.host
```

OAuth 토큰에 대한 모든 요청에는 **<namespace_route>/oauth/authorize**에 대한 요청이 포함됩니다. 대부분의 인증 통합에서는 이 끝점 앞에 인증 프록시를 배치하거나 백업 ID 공급자에 대한 자격 증명의 유효성을 검사하도록 OpenShift Container Platform을 구성합니다. **<namespace_route>/oauth/authorize**에 대한 요청은 CLI와 같은 대화형 로그인 페이지를 표시할 수 없는 사용자 에이전트에서 발생할 수 있습니다. 따라서 OpenShift Container Platform은 대화형 로그인 flows 외에도 **WWW-Authenticate** 챌린지를 사용한 인증을 지원합니다.

인증 프록시를 **<namespace_route>/oauth/authorize** 끝점 앞에 배치하면 대화형 로그인 페이지를 표시하거나 대화형 로그인 flows로 리디렉션하는 대신 인증되지 않은 브라우저 이외의 사용자 에이전트 **WWW-Authenticate** 챌린지를 보냅니다.

참고

브라우저 클라이언트에 대한 CSRF(Cross-Site Request Forgery) 공격을 방지하려면 **X-CSRF-Token** 헤더가 요청에 있는 경우에만 기본 인증 챌린지를 보냅니다. 기본 **WWW-Authenticate** 챌린지를 받을 것으로 예상되는 클라이언트는 이 헤더를 비어 있지 않은 값으로 설정해야 합니다.

인증 프록시가 **WWW-Authenticate** 챌린지를 지원할 수 없거나 OpenShift Container Platform이 WWW-Authenticate 챌린지를 지원하지 않는 ID 공급자를 사용하도록 구성된 경우, 브라우저를 사용하여 **<namespace_route>/oauth/token/request**에서 수동으로 토큰을 가져와야 합니다.

2.3.1.2. API 가장

OpenShift Container Platform API에 대한 요청을 다른 사용자가 보낸 것처럼 작동하도록 구성할 수 있습니다. 자세한 내용은 쿠버네티스 설명서의 사용자 가장을 참조하십시오.

2.3.1.3. Prometheus의 인증 지표

OpenShift Container Platform은 인증 시도 중 다음과 같은 Prometheus 시스템 지표를 캡처합니다.

- **openshift_auth_basic_password_count**는 **oc login** 사용자 이름 및 암호 시도 횟수를 계산합니다.
- **openshift_auth_basic_password_count_result**는 **oc login** 사용자 이름 및 암호 시도 횟수를 결과, **success** 또는 **error**별로 계산합니다.

- **openshift_auth_form_password_count**는 웹 콘솔 로그인 시도 횟수를 계산합니다.
- **openshift_auth_form_password_count_result**는 웹 콘솔 로그인 시도 횟수를 **success** 또는 **error** 등 결과별로 계산합니다.
- **openshift_auth_password_total**은 총 **oc login** 및 웹 콘솔 로그인 시도 횟수를 계산합니다.

3장. 내부 OAUTH 서버 구성

3.1. OPENSHIFT CONTAINER PLATFORM OAUTH 서버

OpenShift Container Platform 마스터에는 내장 OAuth 서버가 포함되어 있습니다. 사용자는 API 인증을 위해 OAuth 액세스 토큰을 가져옵니다.

사용자가 새 OAuth 토큰을 요청하면 OAuth 서버는 구성된 ID 공급자를 사용하여 요청한 사람의 ID를 확인합니다.

그런 다음 해당 ID와 매핑되는 사용자를 결정하고 그 사용자를 위한 액세스 토큰을 만들어 제공합니다.

3.2. OAUTH 토큰 요청 FLOWS 및 응답

OAuth 서버는 표준 인증 코드 부여 및 암시적 부여 OAuth 인증 flows를 지원합니다.

WWW-Authenticate challenges (예: **openshift-challenging-client**)를 요청하기 위해 구성된 client_id와 함께 암시적 부여 flow(**response_type=token**)를 사용하여 OAuth 토큰을 요청하는 경우, **/oauth/authorize**에서 제공 가능한 서버 응답 및 처리 방법은 다음과 같습니다.

상태	내용	클라이언트 응답
302	URL 조각에 **access_token** 매개변수를 포함하는 **Location** 헤더(RFC 6749 섹션 4.2.2)	**access_token** 값을 OAuth 토큰으로 사용하십시오.
302	**error** 쿼리 매개변수를 포함하는 **Location** 헤더(RFC 6749 섹션 4.1.2.1)	실패, **error**(및 선택적 **error_description**) 쿼리 값을 사용자에게 선택적으로 표시합니다.
302	기타 **Location** 헤더	리디렉션을 따르고 해당 규칙을 사용하여 결과를 처리하십시오.
401	**WWW-Authenticate** 헤더 있음	유형이 인식되면(예: **Basic**, **Negotiate**) 챌린지에 응답하고 요청을 다시 제출한 후 해당 규칙을 사용하여 결과를 처리하십시오.
401	**WWW-Authenticate** 헤더 없음	인증할 수 있는 챌린지가 없습니다. 실패했으며 응답 본문(OAuth 토큰을 가져올 수 있는 링크 및 대체 방법에 대한 세부 사항이 포함될 수 있음)을 표시합니다.
기타	기타	실패, 사용자에게 응답 본문을 선택적으로 표시합니다.

3.3. 내부 OAUTH 서버 옵션

내부 OAuth 서버에는 다양한 구성 옵션을 사용할 수 있습니다.

3.3.1. OAuth 토큰 기간 옵션

내부 OAuth 서버는 두 종류의 토큰을 생성합니다.

토큰	설명
액세스 토큰	API에 대한 액세스 권한을 부여하는 장기 토큰입니다.
코드 인증	액세스 토큰으로 교환하는 용도로만 사용하는 단기 토큰입니다.

두 가지 토큰 모두에 대해 기본 기간을 구성할 수 있습니다. 필요한 경우 **OAuthClient** 오브젝트 정의를 사용하여 액세스 토큰 지속 기간을 덮어쓸 수 있습니다.

3.3.2. OAuth 부여 옵션

OAuth 서버에서 사용자가 이전에 권한을 부여하지 않은 클라이언트에 대한 토큰 요청을 수신하는 경우, OAuth 서버에서 수행하는 조치는 OAuth 클라이언트의 권한 부여 전략에 따라 결정됩니다.

토큰을 요청하는 OAuth 클라이언트는 고유의 부여 전략을 제공해야 합니다.

다음과 같은 기본 방법을 적용할 수 있습니다.

부여 옵션	설명
auto	부여를 자동 승인하고 요청을 다시 시도합니다.
prompt	사용자에게 부여를 승인하거나 거부하도록 요청합니다.

3.4. 내부 OAUTH 서버의 토큰 기간 구성

내부 OAuth 서버의 토큰 기간에 대한 기본 옵션을 구성할 수 있습니다.

중요

기본적으로 토큰은 24시간 동안만 유효합니다. 이 시간이 지나면 기존 세션이 만료됩니다.

기본 시간이 충분하지 않으면 다음 절차를 사용하여 수정할 수 있습니다.

절차

1. 토큰 기간 옵션이 포함된 구성 파일을 만듭니다. 다음 파일에서는 이 값을 기본값의 두 배인 48시간으로 설정합니다.

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  tokenConfig:
    accessTokenMaxAgeSeconds: 172800 (1)
```

(1) 액세스 토큰의 수명을 제어하려면 **accessTokenMaxAgeSeconds**를 설정하십시오. 기본 수명은 24시간 또는 86400초입니다. 이 속성은 음수일 수 없습니다. 0으로 설정하면 기본 라이프사이클이 사용됩니다.

2. 새 구성 파일을 적용합니다.

참고

기존 OAuth 서버를 업데이트하므로 **oc apply** 명령을 사용하여 변경 사항을 적용해야 합니다.

```
$ oc apply -f </path/to/file.yaml>
```

3. 변경 사항이 적용되는지 확인합니다.

```
$ oc describe oauth.config.openshift.io/cluster
```

출력 예

```
...
Spec:
  Token Config:
    Access Token Max Age Seconds:  172800
...
```

3.5. 내부 OAUTH 서버에 대한 토큰 비활성 타임아웃 구성

설정된 비활성 기간 후에 만료되도록 OAuth 토큰을 구성할 수 있습니다. 기본적으로 토큰 비활성 타임아웃은 설정되어 있지 않습니다.

참고

토큰 비활성 타임아웃이 OAuth 클라이언트에도 구성된 경우 해당 값은 내부 OAuth 서버 구성에 설정된 타임아웃을 덮어씁니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- IDP(ID 공급자)를 구성했습니다.

절차

1. **OAuth** 구성을 업데이트하여 토큰 비활성 타임아웃을 설정합니다.
   a. **OAuth** 오브젝트를 편집합니다.

```
$ oc edit oauth cluster
```

spec.tokenConfig.accessTokenInactivityTimeout 필드를 추가하고 타임아웃 값을 설정합니다.

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
...
```

```
spec:
  tokenConfig:
    accessTokenInactivityTimeout: 400s (1)
```

(1) 적절한 단위를 사용하여 값을 설정합니다(예: 400초는 **400s**, 30분은 **30m**). 허용되는 최소 타임아웃 값은 **300s**입니다.

b. 파일을 저장하여 변경 사항을 적용합니다.

2. OAuth 서버 Pod가 다시 시작되었는지 확인합니다.

```
$ oc get clusteroperators authentication
```

다음 출력에 표시된 것처럼 **PROGRESSING**이 **False**로 표시될 때까지 다음 단계를 진행하지 마십시오.

출력 예

```
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
authentication   4.8.0     True        False         False      145m
```

3. 쿠버네티스 API 서버 pod의 새 버전이 출시되었는지 확인합니다. 이 작업에는 몇 분 정도 걸립니다.

```
$ oc get clusteroperators kube-apiserver
```

다음 출력에 표시된 것처럼 **PROGRESSING**이 **False**로 표시될 때까지 다음 단계를 진행하지 마십시오.

출력 예

```
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
kube-apiserver   4.8.0     True        False         False      145m
```

PROGRESSING에 **True**가 표시되면 몇 분 기다렸다가 다시 시도하십시오.

검증

1. IDP의 ID로 클러스터에 로그인합니다.
2. 명령을 실행하고 성공했는지 확인합니다.
3. ID를 사용하지 않는 경우 구성된 타임아웃보다 대기 시간이 길어집니다. 이 절차의 예에서는 400초 이상 기다립니다.
4. 동일한 ID의 세션에서 명령을 실행해 봅니다.
구성된 타임아웃보다 비활성 기간이 길어 토큰이 만료되므로 이 명령은 실패할 것입니다.

출력 예

```
error: You must be logged in to the server (Unauthorized)
```

3.6. OAUTH 서버 메타데이터

OpenShift Container Platform에서 실행되는 애플리케이션에서 내장 OAuth 서버에 대한 정보를 검색해야 할 수 있습니다. 예를 들어, 수동 구성이 없는 **<namespace_route>**의 주소를 찾아야 할 수도 있습니다. 이러한 작업을 지원하기 위해 OpenShift Container Platform에서는 IETF OAuth 2.0 인증 서버 메타데이터 초안 사양을 구현합니다.

따라서 클러스터 내에서 실행되는 모든 애플리케이션은 ***https://openshift.default.svc/.well-known/oauth-authorization-server***에 **GET** 요청을 발행하여 다음 정보를 가져올 수 있습니다.

```
{
 "issuer": "https://<namespace_route>", (1)
 "authorization_endpoint": "https://<namespace_route>/oauth/authorize", (2)
 "token_endpoint": "https://<namespace_route>/oauth/token", (3)
 "scopes_supported": [ (4)
  "user:full",
  "user:info",
  "user:check-access",
  "user:list-scoped-projects",
  "user:list-projects"
 ],
 "response_types_supported": [ (5)
  "code",
  "token"
 ],
 "grant_types_supported": [ (6)
  "authorization_code",
  "implicit"
 ],
 "code_challenge_methods_supported": [ (7)
  "plain",
  "S256"
 ]
}
```

(1) **https** 체계를 사용하고 쿼리 또는 조각 구성 요소가 없는 URL에 해당하는 권한 부여 서버의 발행자 식별자. 권한 부여 서버에 대한 정보가 포함된 **.well-known** RFC 5785 리소스가 게시되는 위치입니다.

(2) 권한 부여 서버의 권한 부여 끝점 URL. RFC 6749를 참조하십시오.

(3) 권한 서버의 토큰 Endpoint URL입니다. RFC 6749를 참조하십시오.

(4) 이 권한 부여 서버에서 지원하는 OAuth 2.0 RFC 6749 범위 값 목록이 포함된 JSON 배열. 지원되는 모든 범위 값이 제공되는 것은 아닙니다.

(5) 이 권한 부여 서버에서 지원하는 OAuth 2.0 **response_type** 값 목록이 포함된 JSON 배열. 사용된 배열 값은 RFC 7591의 "OAuth 2.0 동적 클라이언트 등록 프로토콜"에서 정의한 **response_types** 매개변수에 사용된 것과 동일합니다.

(6) 이 권한 부여 서버에서 지원하는 OAuth 2.0 부여 유형 값 목록이 포함된 JSON 배열. 사용된 배열 값은 RFC 7591의 **OAuth 2.0 동적 클라이언트 등록 프로토콜**에서 정의한 **grant_types** 매개변수에 사용된 것과 동일합니다.

(7) 이 권한 부여 서버에서 지원하는 PKCE RFC 7636 코드 챌린지 방법 목록이 포함된 JSON 배열. 코드 챌린지 방법 값은 RFC 7636의 4.3절에 정의된 **code_challenge_method** 매개변수에 사용됩니다.

챌린지 방법 값은 RFC 7636의 4.3절에 정의된 **code_challenge_method** 매개변수에 사용됩니다. 유효한 코드 챌린지 방법 값은 IANA **PKCE 코드 챌린지 방법** 레지스트리에 등록된 값입니다. IANA OAuth 매개변수를 참조하십시오.

3.7. OAUTH API 이벤트 문제 해결

경우에 따라 API 서버는 API 마스터 로그에 직접 액세스하지 않고 디버그하기 어려운 **예기치 않은 조건** 오류 메시지를 반환합니다. 근본적인 오류 원인은 인증되지 않은 사용자에게 서버 상태에 대한 정보를 제공하지 않기 위해 의도적으로 숨겨져 있습니다.

이러한 오류 중 일부는 서비스 계정 OAuth 구성 문제와 관련이 있습니다. 이와 같은 문제는 관리자가 아닌 사용자가 볼 수 있는 이벤트에서 발견됩니다. OAuth 중에 **unexpected condition** 서버 오류가 발생하면 **oc get events**를 실행하여 **ServiceAccount**에서 해당 이벤트를 확인하십시오.

다음 예는 적절한 OAuth 리디렉션 URI가 없는 서비스 계정에 대해 경고합니다.

```
$ oc get events | grep ServiceAccount
```

출력 예

```
1m        1m        1        proxy                ServiceAccount                Warning
NoSAOAuthRedirectURIs   service-account-oauth-client-getter
system:serviceaccount:myproject:proxy has no redirectURIs; set serviceaccounts.openshift.io/oauth-
redirecturi.<some-value>=<redirect> or create a dynamic URI using
serviceaccounts.openshift.io/oauth-redirectreference.<some-value>=<reference>
```

oc describe sa/<service_account_name>을 실행하면 지정된 서비스 계정 이름과 관련된 OAuth 이벤트가 보고됩니다.

```
$ oc describe sa/proxy | grep -A5 Events
```

출력 예

```
Events:
  FirstSeen    LastSeen       Count   From                                SubObjectPath   Type          Reason
Message
  ---------    --------       -----   ----                                -------------   --------      ------                -------
  3m           3m             1       service-account-oauth-client-getter                 Warning
NoSAOAuthRedirectURIs   system:serviceaccount:myproject:proxy has no redirectURIs; set
serviceaccounts.openshift.io/oauth-redirecturi.<some-value>=<redirect> or create a dynamic URI
using serviceaccounts.openshift.io/oauth-redirectreference.<some-value>=<reference>
```

다음은 가능한 이벤트 오류 목록입니다.

리디렉션 URI 주석이 없거나 유효하지 않은 URI가 지정되었습니다

```
Reason                  Message
NoSAOAuthRedirectURIs   system:serviceaccount:myproject:proxy has no redirectURIs; set
serviceaccounts.openshift.io/oauth-redirecturi.<some-value>=<redirect> or create a dynamic URI
using serviceaccounts.openshift.io/oauth-redirectreference.<some-value>=<reference>
```

잘못된 경로가 지정되었습니다

-

```
Reason                  Message
NoSAOAuthRedirectURIs   [routes.route.openshift.io "<name>" not found,
system:serviceaccount:myproject:proxy has no redirectURIs; set serviceaccounts.openshift.io/oauth-
redirecturi.<some-value>=<redirect> or create a dynamic URI using
serviceaccounts.openshift.io/oauth-redirectreference.<some-value>=<reference>]
```

유효하지 않은 참조 유형이 지정되었습니다

```
Reason                  Message
NoSAOAuthRedirectURIs   [no kind "<name>" is registered for version "v1",
system:serviceaccount:myproject:proxy has no redirectURIs; set serviceaccounts.openshift.io/oauth-
redirecturi.<some-value>=<redirect> or create a dynamic URI using
serviceaccounts.openshift.io/oauth-redirectreference.<some-value>=<reference>]
```

SA 토큰이 없습니다

```
Reason                  Message
NoSAOAuthTokens         system:serviceaccount:myproject:proxy has no tokens
```

4장. OAUTH 클라이언트 구성

OpenShift Container Platform에서는 기본적으로 여러 OAuth 클라이언트가 생성됩니다. 추가 OAuth 클라이언트를 등록하고 구성할 수도 있습니다.

4.1. 기본 OAUTH 클라이언트

OpenShift Container Platform API를 시작하면 다음 OAuth 클라이언트가 자동으로 생성됩니다.

OAuth 클라이언트	사용법
openshift-browser-client	대화형 로그인을 처리할 수 있는 사용자 에이전트를 사용하여 **<namespace_route>/oauth/token/request** 에서 토큰을 요청합니다. [1]
openshift-challenging-client	**WWW-Authenticate** 챌린지를 처리할 수 있는 사용자 에이전트로 토큰을 요청합니다.

1. **<namespace_route>**는 네임스페이스 경로를 나타냅니다. 다음 명령을 실행하여 확인할 수 있습니다.

```
$ oc get route oauth-openshift -n openshift-authentication -o json | jq .spec.host
```

4.2. 추가 OAUTH 클라이언트 등록

OpenShift Container Platform 클러스터 인증을 관리하기 위해 추가 OAuth 클라이언트가 필요한 경우 하나의 클라이언트를 등록할 수 있습니다.

절차

- 추가 OAuth 클라이언트를 등록하려면 다음을 수행합니다.

```
$ oc create -f <(echo '
kind: OAuthClient
apiVersion: oauth.openshift.io/v1
metadata:
 name: demo 1
secret: "..." 2
redirectURIs:
 - "http://www.example.com/" 3
grantMethod: prompt 4
')
```

1. OAuth 클라이언트의 **name**은 **<namespace_route>/oauth/authorize** 및 **<namespace_route>/oauth/token**에 요청할 때 **client_id** 매개변수로 사용됩니다.
2. **secret**은 **<namespace_route>/oauth/token**에 요청할 때 **client_secret** 매개변수로 사용됩니다.
3. **<namespace_route>/oauth/authorize** 및 **<namespace_route>/oauth/token**에 대한 요청에 지정된 **redirect_uri** 매개변수는 **redirectURIs** 매개변수 값에 나열된 URI 중 하나와 같거

나 접두사로 지정되어야 합니다.

4 **grantMethod**는 이 클라이언트에서 토큰을 요청할 때 사용자가 아직 액세스 권한을 부여받지 않은 경우 수행할 조치를 결정하는 데 사용됩니다. 부여를 자동으로 승인하고 요청을 다시 시도하려면 **auto**를, 사용자에게 부여를 승인할지 또는 거부할지 묻는 메시지를 표시하려면 **prompt**를 지정합니다.

4.3. OAUTH 클라이언트에 대한 토큰 비활성 타임아웃 구성

설정된 비활성 기간이 지나면 OAuth 토큰이 만료되도록 OAuth 클라이언트를 구성할 수 있습니다. 기본적으로 토큰 비활성 타임아웃은 설정되어 있지 않습니다.

참고

토큰 비활성 타임아웃이 내부 OAuth 서버 구성에도 구성된 경우 OAuth 클라이언트에 설정된 타임아웃이 해당 값을 덮어씁니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- IDP(ID 공급자)를 구성했습니다.

절차

- **OAuthClient** 구성을 업데이트하여 토큰 비활성 타임아웃을 설정합니다.
  a. **OAuthClient** 오브젝트를 편집합니다.

```
$ oc edit oauthclient <oauth_client> 1
```

1 **<oauth_client>**를 구성할 OAuth 클라이언트(예: **console**)로 교체합니다.

accessTokenInactivityTimeoutSeconds 필드를 추가하고 타임아웃 값을 설정합니다.

```
apiVersion: oauth.openshift.io/v1
grantMethod: auto
kind: OAuthClient
metadata:
...
accessTokenInactivityTimeoutSeconds: 600 1
```

1 허용되는 최소 타임아웃 값(초)은 **300**입니다.

  b. 파일을 저장하여 변경 사항을 적용합니다.

검증

1. IDP의 ID로 클러스터에 로그인합니다. 방금 구성한 OAuth 클라이언트를 사용해야 합니다.
2. 조치를 수행하고 성공했는지 확인합니다.

3. ID를 사용하지 않는 경우 구성된 타임아웃보다 대기 시간이 길어집니다. 이 절차의 예에서는 600초 이상 기다립니다.

4. 동일한 ID의 세션에서 작업을 수행해 보십시오.
   구성된 타임아웃보다 비활성 기간이 길어 토큰이 만료되므로 이 시도는 실패할 것입니다.

4.4. 추가 리소스

- OAuthClient [oauth.openshift.io/v1]

5장. 사용자 소유 OAUTH 액세스 토큰 관리

사용자는 자체 OAuth 액세스 토큰을 검토하고 더 이상 필요하지 않은 항목을 삭제할 수 있습니다.

5.1. 사용자 소유 OAUTH 액세스 토큰 나열

사용자 소유 OAuth 액세스 토큰을 나열할 수 있습니다. 토큰 이름은 민감하지 않으며 로그인에 사용할 수 없습니다.

절차

- 모든 사용자 소유 OAuth 액세스 토큰 나열:

```
$ oc get useroauthaccesstokens
```

출력 예

```
NAME      CLIENT NAME                CREATED            EXPIRES
REDIRECT URI                                  SCOPES
<token1>  openshift-challenging-client  2021-01-11T19:25:35Z  2021-01-12 19:25:35
+0000 UTC  https://oauth-openshift.apps.example.com/oauth/token/implicit  user:full
<token2>  openshift-browser-client  2021-01-11T19:27:06Z  2021-01-12 19:27:06 +0000
UTC  https://oauth-openshift.apps.example.com/oauth/token/display  user:full
<token3>  console                  2021-01-11T19:26:29Z  2021-01-12 19:26:29 +0000 UTC
https://console-openshift-console.apps.example.com/auth/callback  user:full
```

- 특정 OAuth 클라이언트의 사용자 소유자 OAuth 액세스 토큰 나열:

```
$ oc get useroauthaccesstokens --field-selector=clientName="console"
```

출력 예

```
NAME      CLIENT NAME                CREATED            EXPIRES
REDIRECT URI                                  SCOPES
<token3>  console                  2021-01-11T19:26:29Z  2021-01-12 19:26:29 +0000 UTC
https://console-openshift-console.apps.example.com/auth/callback  user:full
```

5.2. 사용자 소유 OAUTH 액세스 토큰의 세부 정보 보기

사용자 소유 OAuth 액세스 토큰의 세부 정보를 볼 수 있습니다.

절차

- 사용자 소유 OAuth 액세스 토큰의 세부 정보 설명:

```
$ oc describe useroauthaccesstokens <token_name>
```

출력 예

```
Name:                <token_name> 1
Namespace:
```

```
Labels:                 <none>
Annotations:            <none>
API Version:            oauth.openshift.io/v1
Authorize Token:        sha256~Ksckkug-9Fg_RWn_AUysPoIg-_HqmFI9zUL_CgD8wr8
Client Name:            openshift-browser-client (2)
Expires In:             86400 (3)
Inactivity Timeout Seconds:  317 (4)
Kind:                   UserOAuthAccessToken
Metadata:
  Creation Timestamp:  2021-01-11T19:27:06Z
  Managed Fields:
    API Version:  oauth.openshift.io/v1
    Fields Type:  FieldsV1
    fieldsV1:
      f:authorizeToken:
      f:clientName:
      f:expiresIn:
      f:redirectURI:
      f:scopes:
      f:userName:
      f:userUID:
    Manager:      oauth-server
    Operation:    Update
    Time:         2021-01-11T19:27:06Z
  Resource Version:  30535
  Self Link:         /apis/oauth.openshift.io/v1/useroauthaccesstokens/<token_name>
  UID:               f9d00b67-ab65-489b-8080-e427fa3c6181
Redirect URI:        https://oauth-openshift.apps.example.com/oauth/token/display
Scopes:
  user:full (5)
User Name:  <user_name> (6)
User UID:   82356ab0-95f9-4fb3-9bc0-10f1d6a6a345
Events:     <none>
```

(1) 토큰의 sha256 해시인 토큰 이름입니다. 토큰 이름은 민감하지 않으며 로그인에 사용할 수 없습니다.

(2) 토큰이 시작된 위치를 설명하는 클라이언트 이름입니다.

(3) 이 토큰이 만료되기 전 생성 후 값(초)입니다.

(4) OAuth 서버에 대한 토큰 비활성 타임아웃이 설정된 경우 이 토큰을 더 이상 사용할 수 없기 전 생성 후 값(초)입니다.

(5) 이 토큰의 범위입니다.

(6) 이 토큰과 연관된 사용자 이름입니다.

5.3. 사용자 소유 OAUTH 액세스 토큰 삭제

oc logout 명령은 활성 세션에 대해 OAuth 토큰만 무효화합니다. 다음 절차에 따라 더 이상 필요하지 않은 사용자 소유 OAuth 토큰을 삭제할 수 있습니다.

OAuth 액세스 토큰을 삭제하면 토큰을 사용하는 모든 세션에서 사용자를 로그아웃합니다.

절차

- 사용자 소유 OAuth 액세스 토큰 삭제:

```
$ oc delete useroauthaccesstokens <token_name>
```

출력 예

```
useroauthaccesstoken.oauth.openshift.io "<token_name>" deleted
```

6장. ID 공급자 구성 이해

OpenShift Container Platform 마스터에는 내장 OAuth 서버가 포함되어 있습니다. 개발자와 관리자는 OAuth 액세스 토큰을 가져와 API 인증을 수행합니다.

관리자는 클러스터를 설치한 후 ID 공급자를 지정하도록 OAuth를 구성할 수 있습니다.

6.1. OPENSHIFT CONTAINER PLATFORM의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

6.2. 지원되는 ID 공급자

다음 유형의 ID 공급자를 구성할 수 있습니다.

ID 공급자	설명
htpasswd	**htpasswd**를 사용하여 생성된 플랫 파일에 대해 사용자 이름 및 암호의 유효성을 확인하도록 **htpasswd** ID 공급자를 구성합니다.
Keystone	내부 데이터베이스에 사용자를 저장하는 OpenStack Keystone v3 서버와의 공유 인증을 지원하기 위해 OpenShift Container Platform 클러스터를 Keystone과 통합하도록 **keystone** ID 공급자를 구성합니다.
LDAP	단순 바인드 인증을 사용하여 LDAPv3 서버에 대해 사용자 이름 및 암호의 유효성을 확인하도록 **ldap** ID 공급자를 구성합니다.
기본 인증	사용자가 원격 ID 공급자에 대해 검증된 자격 증명을 사용하여 OpenShift Container Platform에 로그인할 수 있도록 **기본 인증** ID 공급자를 구성합니다. 기본 인증은 일반적인 백엔드 통합 메커니즘입니다.
요청 헤더	**X-Remote-User**와 같은 요청 헤더 값에서 사용자를 확인하도록 **요청 헤더** ID 공급자를 구성합니다. 일반적으로 요청 헤더 값을 설정하는 인증 프록시와 함께 사용됩니다.
GitHub 또는 GitHub Enterprise	GitHub 또는 GitHub Enterprise의 OAuth 인증 서버에 대해 사용자 이름 및 암호의 유효성을 확인하도록 **github** ID 공급자를 구성합니다.
GitLab	GitLab.com 또는 기타 GitLab 인스턴스를 ID 공급자로 사용하도록 **gitlab** ID 공급자를 구성합니다.
Google	Google의 OpenID Connect 통합을 사용하여 **google** ID 공급자를 구성합니다.
OpenID Connect	인증 코드 Flow를 사용하여 OpenID Connect ID 공급자와 통합하도록 **oidc** ID 공급자를 구성합니다.

ID 공급자를 정의하면 RBAC를 사용하여 권한을 정의하고 적용 할 수 있습니다.

6.3. KUBEADMIN 사용자 제거

ID 공급자를 정의하고 새 **cluster-admin** 사용자를 만든 다음 **kubeadmin**을 제거하여 클러스터 보안을 강화할 수 있습니다.

주의

다른 사용자가 **cluster-admin**이 되기 전에 이 절차를 수행하는 경우 OpenShift Container Platform을 다시 설치해야 합니다. 이 명령은 취소할 수 없습니다.

사전 요구 사항

- 하나 이상의 ID 공급자를 구성해야 합니다.
- 사용자에게 **cluster-admin** 역할을 추가해야 합니다.
- 관리자로 로그인해야 합니다.

절차

- **kubeadmin** 시크릿을 제거합니다.

```
$ oc delete secrets kubeadmin -n kube-system
```

6.4. ID 공급자 매개변수

다음 매개변수는 모든 ID 공급자에 공통입니다.

매개변수	설명
name	공급자 사용자 이름에 접두어로 공급자 이름을 지정하여 ID 이름을 만듭니다.

매개변수	설명
mappingMethod	사용자가 로그인할 때 새 ID를 사용자에게 매핑하는 방법을 정의합니다. 다음 값 중 하나를 입력하십시오. **claim** 기본값입니다. 사용자에게 ID의 기본 사용자 이름을 프로비저닝합니다. 해당 사용자 이름의 사용자가 이미 다른 ID에 매핑되어 있는 경우 실패합니다. **lookup** 기존 ID, 사용자 ID 매핑 및 사용자를 조회하지만 사용자 또는 ID를 자동으로 프로비저닝하지는 않습니다. 클러스터 관리자는 이를 통해 수동으로 또는 외부 프로세스를 사용하여 ID 및 사용자를 설정할 수 있습니다. 이 방법을 사용하려면 사용자를 수동으로 프로비저닝해야 합니다. **generate** 사용자에게 ID의 기본 사용자 이름을 프로비저닝합니다. 기본 사용자 이름을 가진 사용자가 이미 기존 ID에 매핑되어 있는 경우 고유한 사용자 이름이 생성됩니다. 예를 들면 **myuser2**입니다. OpenShift Container Platform 사용자 이름과 ID 공급자 사용자 이름(예: LDAP 그룹 동기화)이 정확히 일치해야 하는 외부 프로세스와 함께 이 방법을 사용해서는 안 됩니다. **add** 사용자에게 ID의 기본 사용자 이름을 프로비저닝합니다. 해당 사용자 이름을 가진 사용자가 이미 존재하는 경우 ID가 기존 사용자에게 매핑되고 그 사용자의 기존 ID 매핑에 추가됩니다. 동일한 사용자 집합을 식별하고 동일한 사용자 이름에 매핑되는 ID 공급자를 여럿 구성한 경우 필요합니다.

참고

ID 공급자를 추가하거나 변경할 때 **mappingMethod** 매개변수를 **add**로 설정하면 새 공급자의 ID를 기존 사용자에게 매핑할 수 있습니다.

6.5. ID 공급자 CR 샘플

다음 CR(사용자 정의 리소스)에서는 ID 공급자를 구성하는 데 사용되는 매개변수 및 기본값을 보여줍니다. 이 예에서는 htpasswd ID 공급자를 사용합니다.

ID 공급자 CR 샘플

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: my_identity_provider 1
    mappingMethod: claim 2
    type: HTPasswd
    htpasswd:
      fileData:
        name: htpass-secret 3
```

1 이 공급자 이름은 공급자 사용자 이름에 접두어로 지정되어 ID 이름을 형성합니다.

2 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

3 **htpasswd**를 사용하여 생성한 파일이 포함된 기존 시크릿입니다.

7장. ID 공급자 구성

7.1. HTPASSWD ID 공급자 구성

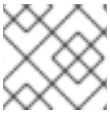
htpasswd ID 공급자를 구성하여 사용자가 htpasswd 파일에서 인증 정보를 사용하여 OpenShift Container Platform에 로그인할 수 있도록 합니다.

htpasswd ID 공급자를 정의하려면 다음 작업을 수행합니다.

1. **htpasswd** 파일을 생성하여 사용자 및 암호 정보를 저장합니다.
2. **htpasswd** 파일을 나타내는 시크릿을 생성합니다.
3. 시크릿 을 참조하는 htpasswd ID 공급자 리소스를 정의합니다 .
4. 리소스를 기본 OAuth 구성에 적용하여 ID 공급자를 추가합니다.

7.1.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.



참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.1.2. htpasswd 인증 정보

OpenShift Container Platform에서 htpasswd 인증을 사용하면 htpasswd 파일을 기반으로 사용자를 식별할 수 있습니다. htpasswd 파일은 각 사용자의 사용자 이름과 해시된 암호를 포함하는 플랫 파일입니다. **htpasswd** 유틸리티를 사용하여 이 파일을 생성할 수 있습니다.

7.1.3. htpasswd 파일 생성

htpasswd 파일을 생성하는 방법에 대한 지침은 다음 섹션 중 하나를 참조하십시오.

- Linux를 사용하여 htpasswd 파일 생성
- Windows를 사용하여 htpasswd 파일 생성

7.1.3.1. Linux를 사용하여 htpasswd 파일 생성

htpasswd ID 공급자를 사용하려면 **htpasswd** 를 사용하여 클러스터의 사용자 이름 및 암호가 포함된 플랫 파일을 생성해야 합니다.

사전 요구 사항

- **htpasswd** 유틸리티에 액세스할 수 있습니다. Red Hat Enterprise Linux에서는 **httpd-tools** 패키지를 설치하면 액세스할 수 있습니다.

절차

1. 사용자 이름과 해시된 암호로 플랫 파일을 생성하거나 업데이트합니다.

```
-
```

```
$ htpasswd -c -B -b </path/to/users.htpasswd> <user_name> <password>
```

이 명령에서는 해시된 버전의 암호를 생성합니다.

예를 들면 다음과 같습니다.

```
$ htpasswd -c -B -b users.htpasswd user1 MyPassword!
```

출력 예

```
Adding password for user user1
```

2. 인증 정보를 파일에 계속 추가하거나 업데이트합니다.

```
$ htpasswd -B -b </path/to/users.htpasswd> <user_name> <password>
```

7.1.3.2. Windows를 사용하여 htpasswd 파일 생성

htpasswd ID 공급자를 사용하려면 **htpasswd** 를 사용하여 클러스터의 사용자 이름 및 암호가 포함된 플랫 파일을 생성해야 합니다.

사전 요구 사항

- **htpasswd.exe**에 액세스할 수 있습니다. 이 파일은 많은 Apache httpd 배포판의 **\bin** 디렉터리에 포함되어 있습니다.

절차

1. 사용자 이름과 해시된 암호로 플랫 파일을 생성하거나 업데이트합니다.

```
> htpasswd.exe -c -B -b <\path\to\users.htpasswd> <user_name> <password>
```

이 명령에서는 해시된 버전의 암호를 생성합니다.

예를 들면 다음과 같습니다.

```
> htpasswd.exe -c -B -b users.htpasswd user1 MyPassword!
```

출력 예

```
Adding password for user user1
```

2. 인증 정보를 파일에 계속 추가하거나 업데이트합니다.

```
> htpasswd.exe -b <\path\to\users.htpasswd> <user_name> <password>
```

7.1.4. htpasswd 시크릿 생성

htpasswd ID 공급자를 사용하려면 htpasswd 사용자 파일이 포함된 시크릿을 정의해야 합니다.

사전 요구 사항

- htpasswd 파일을 생성합니다.

절차

- htpasswd 사용자 파일이 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic htpass-secret --from-file=htpasswd=<path_to_users.htpasswd> -n
openshift-config 1
```

1 **--from-file** 인수에 대한 사용자 파일이 포함된 보안 키의 이름은 위의 명령과 같이 **htpasswd**로 지정해야 합니다.

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: htpass-secret
  namespace: openshift-config
type: Opaque
data:
  htpasswd: <base64_encoded_htpasswd_file_contents>
```

7.1.5. htpasswd CR 샘플

다음 CR(사용자 정의 리소스)에는 htpasswd ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

htpasswd CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: my_htpasswd_provider 1
    mappingMethod: claim 2
    type: HTPasswd
    htpasswd:
      fileData:
        name: htpass-secret 3
```

1 이 공급자 이름은 공급자 사용자 이름에 접두어로 지정되어 ID 이름을 형성합니다.

2 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

3 **htpasswd**를 사용하여 생성한 파일이 포함된 기존 시크릿입니다.

추가 리소스

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.1.6. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```

참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

```
$ oc whoami
```

7.1.7. htpasswd ID 공급자의 사용자 업데이트

기존 htpasswd ID 공급자에서 사용자를 추가하거나 제거할 수 있습니다.

사전 요구 사항

- htpasswd 사용자 파일이 포함된 **Secret** 오브젝트를 생성했습니다. 이 절차에서는 이름을 **htpass-secret**이라고 가정합니다.
- htpasswd ID 공급자를 구성했습니다. 이 절차에서는 이름이 **my_htpasswd_provider**라고 가정합니다.
- **htpasswd** 유틸리티에 액세스할 수 있습니다. Red Hat Enterprise Linux에서는 **httpd-tools** 패키지를 설치하면 액세스할 수 있습니다.
- 클러스터 관리자 권한이 있어야 합니다.

절차

1. **htpass-secret Secret** 오브젝트에서 htpasswd 파일을 검색하여 파일 시스템에 저장합니다.

```
$ oc get secret htpass-secret -ojsonpath={.data.htpasswd} -n openshift-config | base64 --
decode > users.htpasswd
```

2. **users.htpasswd** 파일에서 사용자를 추가하거나 제거합니다.

- 새로운 사용자 추가:

```
$ htpasswd -bB users.htpasswd <username> <password>
```

출력 예

```
Adding password for user <username>
```

- 기존 사용자 제거:

```
$ htpasswd -D users.htpasswd <username>
```

출력 예

```
Deleting password for user <username>
```

3. **users.htpasswd** 파일에서 **htpass-secret Secret** 오브젝트를 업데이트된 사용자로 교체합니다.

```
$ oc create secret generic htpass-secret --from-file=htpasswd=users.htpasswd --dry-
run=client -o yaml -n openshift-config | oc replace -f -
```

작은 정보

다음 YAML을 적용하여 시크릿을 대체할 수 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: htpass-secret
  namespace: openshift-config
type: Opaque
data:
  htpasswd: <base64_encoded_htpasswd_file_contents>
```

4. 하나 이상의 사용자를 제거한 경우 각 사용자에 대한 기존 리소스를 추가로 제거해야 합니다.

a. **User** 오브젝트 삭제:

```
$ oc delete user <username>
```

출력 예

```
user.user.openshift.io "<username>" deleted
```

사용자를 제거해야 합니다. 그러지 않으면 토큰이 만료되지 않는 한 사용자가 토큰을 계속 사용할 수 있습니다.

b. 사용자의 **Identity** 오브젝트 삭제:

```
$ oc delete identity my_htpasswd_provider:<username>
```

출력 예

```
identity.user.openshift.io "my_htpasswd_provider:<username>" deleted
```

7.1.8. 웹 콘솔을 사용하여 ID 공급자 구성

CLI 대신 웹 콘솔을 통해 ID 공급자(IDP)를 구성합니다.

사전 요구 사항

- 웹 콘솔에 클러스터 관리자로 로그인해야 합니다.

절차

1. **관리 → 클러스터 설정**으로 이동합니다.
2. **글로벌 구성** 탭에서 **OAuth**를 클릭합니다.
3. **ID 공급자** 섹션의 **추가** 드롭다운 메뉴에서 ID 공급자를 선택합니다.

참고

기존 IDP를 덮어쓰지 않고 웹 콘솔을 통해 여러 IDP를 지정할 수 있습니다.

7.2. KEYSTONE ID 공급자 구성

내부 데이터베이스에 사용자를 저장하는 OpenStack Keystone v3 서버와의 공유 인증을 지원하기 위해 OpenShift Container Platform 클러스터를 Keystone과 통합하도록 **keystone** ID 공급자를 구성합니다. 이 구성을 통해 사용자는 Keystone 자격 증명을 사용하여 OpenShift Container Platform에 로그인할 수 있습니다.

7.2.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.2.2. Keystone 인증 정보

Keystone은 ID, 토큰, 카탈로그 및 정책 서비스를 제공하는 OpenStack 프로젝트입니다.

새 OpenShift Container Platform 사용자가 Keystone 사용자 이름 또는 고유한 Keystone ID를 기반으로 하도록 Keystone과의 통합을 구성할 수 있습니다. 두 방법 모두 사용자는 Keystone 사용자 이름과 암호를 입력하여 로그인합니다. Keystone ID에서 OpenShift Container Platform 사용자를 배치하는 것은 Keystone 사용자를 삭제하고 해당 사용자 이름으로 새 Keystone 사용자를 만드는 경우 새 사용자가 이전 사용자의 리소스에 액세스할 수 있기 때문에 더 안전합니다.

7.2.3. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 키 및 인증서가 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret tls <secret_name> --key=key.pem --cert=cert.pem -n openshift-config
```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: kubernetes.io/tls
data:
  tls.crt: <base64_encoded_cert>
  tls.key: <base64_encoded_key>
```

7.2.4. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.2.5. Keystone CR 샘플

다음 CR(사용자 정의 리소스)에는 Keystone ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

Keystone CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: keystoneidp 1
    mappingMethod: claim 2
    type: Keystone
    keystone:
      domainName: default 3
      url: https://keystone.example.com:5000 4
      ca: 5
        name: ca-config-map
      tlsClientCert: 6
        name: client-cert-secret
      tlsClientKey: 7
        name: client-key-secret
```

1. 이 공급자 이름은 공급자 사용자 이름에 접두어로 지정되어 ID 이름을 형성합니다.
2. 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.
3. Keystone 도메인 이름. Keystone에서는 사용자 이름이 도메인에 따라 다릅니다. 단일 도메인만 지원됩니다.
4. Keystone 서버 연결에 사용하는 URL입니다(필수). https를 사용해야 합니다.
5. 선택 사항: OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다.
6. 선택 사항: OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, 구성된 URL에 요청할 때 제공할 클라이언트 인증서가 포함됩니다.

7 OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, 클라이언트 인증서에 대한 키가 포함됩니다. **tlsClientCert**가 지정된 경우 필수입니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.2.6. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```

참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

```
$ oc whoami
```

7.3. LDAP ID 공급자 구성

단순 바인드 인증을 사용하여 LDAPv3 서버에 대해 사용자 이름 및 암호의 유효성을 확인하도록 **ldap** ID 공급자를 구성합니다.

7.3.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.3.2. LDAP 인증 정보

LDAP 디렉터리는 인증 중 제공된 사용자 이름과 일치하는 항목으로 검색됩니다. 고유한 일치 항목이 1개 발견되는 경우 해당 항목의 고유 이름(DN)과 제공된 암호를 사용하여 단순 바인딩이 시도됩니다.

다음 단계를 수행합니다.

1. 구성된 **URL**의 속성 및 필터를 사용자가 입력한 사용자 이름과 결합하여 검색 필터를 생성합니다.
2. 생성된 필터를 사용하여 디렉터리를 검색합니다. 검색에서 정확히 하나의 항목을 반환하지 않으면 액세스를 거부합니다.
3. 검색에서 검색된 항목의 DN과 사용자 제공 암호를 사용하여 LDAP 서버에 바인딩합니다.
4. 바인딩에 실패하면 액세스를 거부합니다.
5. 바인딩이 성공하면 구성된 속성을 ID, 이메일 주소, 표시 이름, 기본 사용자 이름으로 사용하여 ID를 빌드합니다.

구성된 **URL**은 RFC 2255 URL이며, 사용할 LDAP 호스트 및 검색 매개변수를 지정합니다. URL 구문은 다음과 같습니다.

```
ldap://host:port/basedn?attribute?scope?filter
```

이 URL의 경우

URL 구성 요소	설명
ldap	일반 LDAP의 경우 **ldap** 문자열을 사용합니다. 보안 LDAP(LDAPS)의 경우 대신**ldaps**를 사용합니다.
host:port	LDAP 서버의 이름 및 포트입니다. ldap의 경우 기본값은 **localhost:389** 이고 LDAPS의 경우 **localhost:636**입니다.
basedn	모든 검색을 시작해야 하는 디렉터리 분기의 DN입니다. 적어도 디렉터리 트리의 맨 위에 있어야 하지만 디렉터리에 하위 트리를 지정할 수도 있습니다.
attribute	검색할 속성입니다. RFC 2255에서는 쉼표로 구분된 속성 목록을 사용할 수 있지만 제공되는 속성 수와 관계없이 첫 번째 속성만 사용됩니다. 속성이 제공되지 않는 경우 기본값은 **uid**를 사용하는 것입니다. 사용할 하위 트리의 모든 항목에서 고유한 속성을 선택하는 것이 좋습니다.
scope	검색 범위입니다. **one** 또는**sub**일 수 있습니다. 범위가 제공되지 않는 경우 기본값은**sub** 범위를 사용하는 것입니다.
filter	유효한 LDAP 검색 필터입니다. 제공하지 않는 경우 기본값은 **(objectClass=*)**입니다.

검색을 수행할 때 속성, 필터, 제공된 사용자 이름을 결합하여 다음과 같은 검색 필터가 생성됩니다.

```
(&(<filter>)(<attribute>=<username>))
```

예를 들어 다음과 같은 URL을 살펴보십시오.

```
ldap://ldap.example.com/o=Acme?cn?sub?(enabled=true)
```

클라이언트가 사용자 이름 **bob**을 사용하여 연결을 시도하는 경우 결과 검색 필터는 **(&(enabled=true)(cn=bob))**입니다.

LDAP 디렉터리에서 검색에 인증이 필요한 경우 항목을 검색하는 데 사용할 **bindDN** 및 **bindPassword**를 지정하십시오.

7.3.3. LDAP 시크릿 생성

ID 공급자를 사용하려면 **bindPassword** 필드가 포함된 OpenShift Container Platform **Secret** 오브젝트를 정의해야 합니다.

절차

- **bindPassword** 필드가 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic ldap-secret --from-literal=bindPassword=<secret> -n openshift-config 1
```

1 **--from-literal** 인수에 대한 bindPassword를 포함하는 시크릿 키를 **bindPassword**라고 해야 합니다.

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: ldap-secret
  namespace: openshift-config
type: Opaque
data:
  bindPassword: <base64_encoded_bind_password>
```

7.3.4. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

-

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.3.5. LDAP CR 샘플

다음 CR(사용자 정의 리소스)에는 LDAP ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

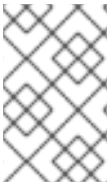
LDAP CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: ldapidp (1)
    mappingMethod: claim (2)
    type: LDAP
    ldap:
      attributes:
        id: (3)
        - dn
        email: (4)
        - mail
        name: (5)
        - cn
        preferredUsername: (6)
        - uid
      bindDN: "" (7)
      bindPassword: (8)
        name: ldap-secret
      ca: (9)
        name: ca-config-map
      insecure: false (10)
      url: "ldap://ldap.example.com/ou=users,dc=acme,dc=com?uid" (11)
```

(1) 이 공급자 이름은 반환된 사용자 ID 앞에 접두어로 지정되어 ID 이름을 형성합니다.

(2) 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

3. ID로 사용할 속성 목록입니다. 비어 있지 않은 첫 번째 속성이 사용됩니다. 하나 이상의 속성이 필요합니다. 나열된 어떤 속성에도 값이 없는 경우 인증이 실패합니다. 정의된 속성은 raw로 검색되므로
4. 이메일 주소로 사용할 속성 목록입니다. 비어 있지 않은 첫 번째 속성이 사용됩니다.
5. 표시 이름으로 사용할 속성 목록입니다. 비어 있지 않은 첫 번째 속성이 사용됩니다.
6. 이 ID에 대해 사용자를 프로비저닝할 때 기본 사용자 이름으로 사용할 속성 목록입니다. 비어 있지 않은 첫 번째 속성이 사용됩니다.
7. 검색 단계에서 바인딩하는 데 사용할 선택적 DN입니다. **bindPassword**가 정의된 경우 설정해야 합니다.
8. 바인딩 암호가 포함된 OpenShift Container Platform **Secret** 오브젝트에 대한 선택적 참조입니다. **bindDN**이 정의된 경우 설정해야 합니다.
9. 선택 사항: OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다. **insecure**가 **false**인 경우에만 사용됩니다.
10. **true**인 경우 서버에 TLS 연결이 이루어지지 않습니다. **false**인 경우 **ldaps://** URL은 TLS를 사용하여 연결되고, **ldap://** URL은 TLS로 업그레이드됩니다. **ldaps://** URL을 사용 중인 경우 이 URL은 항상 TLS를 사용하여 연결을 시도하므로 **false**로 설정해야 합니다.
11. 사용할 LDAP 호스트 및 검색 매개변수를 지정하는 RFC 2255 URL입니다.



참고

LDAP 통합을 위해 사용자를 허용 목록에 추가하려면 **lookup** 매핑 방법을 사용하십시오. LDAP에서 로그인하려면 클러스터 관리자가 각 LDAP 사용자에 대한 **Identity** 오브젝트 및 **User** 오브젝트를 생성해야 합니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.3.6. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

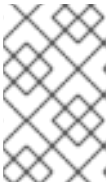
사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```



참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

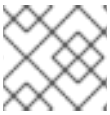
```
$ oc whoami
```

7.4. 기본 인증 ID 공급자 구성

사용자가 원격 ID 공급자에 대해 검증된 자격 증명을 사용하여 OpenShift Container Platform에 로그인할 수 있도록 **기본 인증** ID 공급자를 구성합니다. 기본 인증은 일반적인 백엔드 통합 메커니즘입니다.

7.4.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.



참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.4.2. 기본 인증 정보

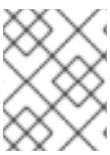
기본 인증은 일반 백엔드 통합 메커니즘으로, 사용자가 원격 ID 공급자에 대해 검증된 인증 정보를 사용하여 OpenShift Container Platform에 로그인할 수 있습니다.

기본 인증은 일반적이므로 이 ID 공급자를 고급 인증 구성에 사용할 수 있습니다.

중요

기본 인증은 원격 서버에 대한 HTTPS 연결을 통해 잠재적인 사용자 ID 및 암호 스누핑 및 중간자 공격을 방지해야 합니다.

기본 인증이 구성된 상태에서 사용자가 OpenShift Container Platform에 사용자 이름과 암호를 보내면 서버 간 요청을 통해 원격 서버에 대해 해당 자격 증명의 유효성을 확인하고, 자격 증명을 기본 인증 헤더로 전달합니다. 이를 위해서는 사용자가 로그인하는 동안 사용자의 자격 증명을 OpenShift Container Platform에 보내야 합니다.



참고

이 작업은 사용자 이름/암호 로그인 메커니즘에서만 작동하며, OpenShift Container Platform에서 원격 인증 서버에 네트워크를 요청할 수 있어야 합니다.

사용자 이름 및 암호는 기본 인증으로 보호되고 JSON이 반환되는 원격 URL에 대해 유효성이 확인됩니다.

401 응답은 인증 실패를 나타냅니다.

200 이외의 상태 또는 비어 있지 않은 "error" 키는 오류를 나타냅니다.

```
{"error":"Error message"}
```

sub(제목) 키가 **200**인 태는 실행 성공을 나타냅니다.

```
{"sub":"userid"} 1
```

1 제목은 인증된 사용자에게 고유해야 하며 수정할 수 없어야 합니다.

성공적인 응답은 선택적으로 다음과 같은 추가 데이터를 제공할 수 있습니다.

- **name** 키를 사용하는 표시 이름. 예를 들면 다음과 같습니다.

  ```
  {"sub":"userid", "name": "User Name", ...}
  ```

- **email** 키를 사용하는 이메일 주소. 예를 들면 다음과 같습니다.

  ```
  {"sub":"userid", "email":"user@example.com", ...}
  ```

- **preferred_username** 키를 사용하는 기본 사용자 이름. 변경 불가능한 고유 주체가 데이터베이스 키 또는 UID이고 더 읽기 쉬운 이름이 존재하는 경우 유용합니다. 인증된 ID에 대해 OpenShift Container Platform 사용자를 프로비저닝할 때 힌트로 사용됩니다. 예를 들면 다음과 같습니다.

  ```
  {"sub":"014fbff9a07c", "preferred_username":"bob", ...}
  ```

7.4.3. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 키 및 인증서가 포함된 **Secret** 오브젝트를 생성합니다.

  ```
  $ oc create secret tls <secret_name> --key=key.pem --cert=cert.pem -n openshift-config
  ```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: kubernetes.io/tls
data:
  tls.crt: <base64_encoded_cert>
  tls.key: <base64_encoded_key>
```

7.4.4. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.4.5. 기본 인증 CR 샘플

다음 CR(사용자 정의 리소스)에는 기본 인증 ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

기본 인증 CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
```

```
- name: basicidp (1)
  mappingMethod: claim (2)
  type: BasicAuth
  basicAuth:
    url: https://www.example.com/remote-idp (3)
    ca: (4)
      name: ca-config-map
    tlsClientCert: (5)
      name: client-cert-secret
    tlsClientKey: (6)
      name: client-key-secret
```

(1) 이 공급자 이름은 반환된 사용자 ID 앞에 접두어로 지정되어 ID 이름을 형성합니다.

(2) 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

(3) 기본 인증 헤더에서 인증 정보를 수락하는 URL.

(4) 선택 사항: OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다.

(5) 선택 사항: OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, 구성된 URL에 요청할 때 제공할 클라이언트 인증서가 포함됩니다.

(6) OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, 클라이언트 인증서에 대한 키가 포함됩니다. **tlsClientCert**가 지정된 경우 필수입니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.4.6. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```

참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

```
$ oc whoami
```

7.4.7. 기본 ID 공급자에 대한 Apache HTTPD 구성 예

OpenShift Container Platform 4의 기본 ID 공급자(IDP) 구성에서는 IDP 서버에서 JSON을 사용하여 성공 및 실패로 응답해야 합니다. Apache HTTPD에서 CGI 스크립팅을 사용하여 이러한 작업을 수행할 수 있습니다. 이 섹션에서는 예를 제공합니다.

예: /etc/httpd/conf.d/login.conf

```
<VirtualHost *:443>
  # CGI Scripts in here
  DocumentRoot /var/www/cgi-bin

  # SSL Directives
  SSLEngine on
  SSLCipherSuite PROFILE=SYSTEM
  SSLProxyCipherSuite PROFILE=SYSTEM
  SSLCertificateFile /etc/pki/tls/certs/localhost.crt
  SSLCertificateKeyFile /etc/pki/tls/private/localhost.key

  # Configure HTTPD to execute scripts
  ScriptAlias /basic /var/www/cgi-bin

  # Handles a failed login attempt
  ErrorDocument 401 /basic/fail.cgi

  # Handles authentication
  <Location /basic/login.cgi>
    AuthType Basic
    AuthName "Please Log In"
    AuthBasicProvider file
    AuthUserFile /etc/httpd/conf/passwords
    Require valid-user
  </Location>
</VirtualHost>
```

예: /var/www/cgi-bin/login.cgi

```
#!/bin/bash
```

```
echo "Content-Type: application/json"
echo ""
echo '{"sub":"userid", "name":"'$REMOTE_USER'"}'
exit 0
```

예: /var/www/cgi-bin/fail.cgi

```
#!/bin/bash
echo "Content-Type: application/json"
echo ""
echo '{"error": "Login failure"}'
exit 0
```

7.4.7.1. 파일 요구 사항

다음은 Apache HTTPD 웹 서버에서 생성하는 파일에 대한 요구 사항입니다.

- **login.cgi** 및 **fail.cgi**를 실행할 수 있어야 합니다(**chmod +x**).
- SELinux가 활성화되어 있는 경우 **login.cgi** 및 **fail.cgi**에 적절한 SELinux 컨텍스트 **restorecon -RFv /var/www/cgi-bin**가 있어야 합니다. 그러지 않으면 **ls -laZ**를 사용하여 컨텍스트가 **httpd_sys_script_exec_t**인지 확인합니다.
- **login.cgi**는 사용자가 **Require and Auth** 지시문에 따라 성공적으로 로그인한 경우에만 실행됩니다.
- 사용자가 로그인하지 못하면 **fail.cgi**가 실행되어 **HTTP 401** 응답이 표시됩니다.

7.4.8. 기본 인증 문제 해결

가장 일반적인 문제는 백엔드 서버에 대한 네트워크 연결과 관련이 있습니다. 간단한 디버깅을 위해 마스터에서 **curl** 명령을 실행합니다. 성공적인 로그인을 테스트하려면 다음 예제 명령에서 **<user>** 및 **<password>**를 유효한 자격 증명으로 교체하십시오. 잘못된 로그인을 테스트하려면 잘못된 인증 정보로 대체합니다.

```
$ curl --cacert /path/to/ca.crt --cert /path/to/client.crt --key /path/to/client.key -u <user>:<password> -v
https://www.example.com/remote-idp
```

성공적인 응답

sub(제목) 키가 **200**인 태는 실행 성공을 나타냅니다.

```
{"sub":"userid"}
```

제목은 인증된 사용자에게 고유해야 하며 수정할 수 없어야 합니다.

성공적인 응답은 선택적으로 다음과 같은 추가 데이터를 제공할 수 있습니다.

- **name** 키를 사용하는 표시 이름:

  ```
  {"sub":"userid", "name": "User Name", ...}
  ```

- **email** 키를 사용하는 이메일 주소:

```
{"sub":"userid", "email":"user@example.com", ...}
```

- **preferred_username** 키를 사용하는 기본 사용자 이름:

```
{"sub":"014fbff9a07c", "preferred_username":"bob", ...}
```

preferred_username은 변경 불가능한 고유한 제목이 데이터베이스 키 또는 UID이고 더 읽기 쉬운 이름이 존재하는 경우 유용합니다. 인증된 ID에 대해 OpenShift Container Platform 사용자를 프로비저닝할 때 힌트로 사용됩니다.

실패한 응답

- **401** 응답은 인증 실패를 나타냅니다.
- **200** 이외의 상태 또는 비어 있지 않은 "error" 키는 오류(**{"error":"Error message"}**)를 나타냅니다.

7.5. 요청 헤더 ID 공급자 구성

X-Remote - User 와 같은 요청 헤더 값에서 사용자를 식별하도록 요청 헤더 ID 공급자를 구성합니다. 일반적으로 요청 헤더 값을 설정하는 인증 프록시와 함께 사용됩니다.

7.5.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.5.2. 요청 헤더 인증 정보

요청 헤더 ID 공급자는 **X-Remote-User**와 같은 요청 헤더 값에서 사용자를 확인합니다. 일반적으로 요청 헤더 값을 설정하는 인증 프록시와 함께 사용됩니다. 요청 헤더 ID 공급자는 htpasswd, Keystone, LDAP 또는 기본 인증과 같이 직접 암호 로그인을 사용하는 다른 ID 공급자와 결합할 수 없습니다.

참고

커뮤니티 지원 SAML 인증과 같은 고급 구성에도 요청 헤더 ID 공급자를 사용할 수 있습니다. 이 솔루션은 Red Hat에서 지원하지 않습니다.

사용자가 이 ID 공급자를 사용하여 인증하려면 인증 프록시를 통해 **https://*<namespace_route>*/oauth/authorize**(및 하위 경로)에 액세스해야 합니다. 이를 위해서는 **https://*<namespace_route>*/oauth/authorize**로 프록시하는 프록시 끝점으로 OAuth 토큰에 대한 인증되지 않은 요청을 리디렉션하도록 OAuth 서버를 구성해야 합니다.

브라우저 기반 로그인 flows가 필요한 클라이언트의 인증되지 않은 요청을 리디렉션하려면 다음을 수행합니다.

- **provider.loginURL** 매개변수를 대화형 클라이언트를 인증하는 인증 프록시 URL로 설정한 다음 요청을 **https://*<namespace_route>*/oauth/authorize**로 프록시합니다.

WWW 인증 챌린지가 예상되는 클라이언트의 인증되지 않은 요청을 리디렉션하려면 다음을 수행합니다.

- **provider.challengeURL** 매개변수를 **WWW 인증** 챌린지가 예상되는 클라이언트를 인증하는 인증 프록시 URL로 설정한 후 요청을 **https://*<namespace_route>*/oauth/authorize**로 프록시합니다.

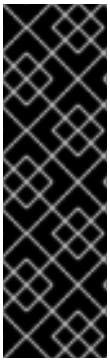
provider.challengeURL 및 **provider.loginURL** 매개변수는 URL 조회 부분에 다음 토큰을 포함할 수 있습니다.

- **${url}**은 현재 URL로 교체되며 쿼리 매개변수에서 안전하도록 이스케이프됩니다.
  예: **https://www.example.com/sso-login?then=${url}**
- **${query}**는 이스케이프 처리되지 않은 현재 쿼리 문자열로 교체됩니다.
  예: **https://www.example.com/auth-proxy/oauth/authorize?${query}**

중요

OpenShift Container Platform 4.1부터는 프록시에서 상호 TLS를 지원해야 합니다.

7.5.2.1. Microsoft Windows에서 SSPI 연결 지원



중요

Microsoft Windows에서 SSPI 연결 지원을 사용하는 것은 기술 프리뷰 기능입니다. 기술 프리뷰 기능은 Red Hat 프로덕션 서비스 수준 계약(SLA)에서 지원하지 않으며, 기능상 완전하지 않을 수 있어 프로덕션에 사용하지 않는 것이 좋습니다. 이러한 기능을 사용하면 향후 제품 기능을 조기에 이용할 수 있어 개발 과정에서 고객이 기능을 테스트하고 피드백을 제공할 수 있습니다.

Red Hat 기술 프리뷰 기능 지원 범위에 대한 자세한 내용은 https://access.redhat.com/support/offerings/techpreview/를 참조하십시오.

OpenShift CLI **oc**는 Microsft Windows에서 SSO 흐름을 허용하기 위해 SSPI(Security Support Provider Interface)를 지원합니다. 요청 헤더 ID 공급자를 GSSAPI 지원 프록시와 함께 사용하여 Active Directory 서버를 OpenShift Container Platform에 연결하는 경우, 사용자가 도메인에 가입된 Microsoft Windows 컴퓨터에서 **oc** 명령줄 인터페이스를 사용하여 OpenShift Container Platform을 자동으로 인증할 수 있습니다.

7.5.3. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.5.4. 요청 헤더 CR 샘플

다음 CR(사용자 정의 리소스)에는 요청 헤더 ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

요청 헤더 CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: requestheaderidp 1
    mappingMethod: claim 2
    type: RequestHeader
    requestHeader:
      challengeURL: "https://www.example.com/challenging-proxy/oauth/authorize?${query}" 3
      loginURL: "https://www.example.com/login-proxy/oauth/authorize?${query}" 4
      ca: 5
        name: ca-config-map
      clientCommonNames: 6
      - my-auth-proxy
      headers: 7
      - X-Remote-User
      - SSO-User
      emailHeaders: 8
      - X-Remote-User-Email
      nameHeaders: 9
      - X-Remote-User-Display-Name
      preferredUsernameHeaders: 10
      - X-Remote-User-Login
```

1. 이 공급자 이름은 요청 헤더에서 사용자 이름 앞에 접두어로 지정되어 ID 이름을 형성합니다.
2. 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.
3. 선택 사항: 인증되지 않은 **/oauth/authorize** 요청을 로 리디렉션하는 URL로, 브라우저 기반 클라이언트를 인증한 다음 해당 요청을 **https://*<namespace_route>*/oauth/authorize로 프록시합니다.** OAuth 승인 흐름이 제대로 작동하려면 **https://*<namespace_route>*/oauth/authorize 로 프록시되**

는 URL은 **/authorize** (후행 슬래시 없음) 및 프록시 하위 경로로 끝나야 합니다. **${url}** 은 현재 URL로 교체되고 쿼리 매개변수에서 안전하도록 이스케이프됩니다. **${query}** 는 현재 쿼리 문자열로 교체됩니다. 이 속성이 정의되어 있지 않으면 **loginURL**을 사용해야 합니다.

4 선택 사항: 인증되지 않은 **/oauth/authorize** 요청을 리디렉션할 URL로, **WWW-Authenticate 챌린지**를 예상하는 클라이언트를 인증한 다음 **https://*<namespace_route>*/oauth/authorize로 프록시합니다.${url}** 은 현재 URL로 교체되고 쿼리 매개 변수에서 안전하도록 이스케이프됩니다. **${query}** 는 현재 쿼리 문자열로 교체됩니다. 이 속성이 정의되지 않은 경우 **challengeURL**을 사용해야 합니다.

5 PEM 인코딩 인증서 번들이 포함된 OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조입니다. 원격 서버에서 제공하는 TLS 인증서의 유효성을 확인하는 신뢰 앵커로 사용됩니다.

중요

OpenShift Container Platform 4.1부터는 이 ID 공급자에 **ca** 필드가 있어야 합니다. 이는 프록시에서 상호 TLS를 지원해야 함을 의미합니다.

6 선택 사항: 공통 이름(**cn**) 목록. 설정된 경우 요청 헤더에서 사용자 이름을 확인하기 전에 지정된 목록에 공통 이름(**cn**)이 있는 유효한 클라이언트 인증서를 제공해야 합니다. 비어 있는 경우 모든 공통 이름이 허용됩니다. **ca**와 함께만 사용할 수 있습니다.

7 사용자 ID에 대해 순서대로 확인할 헤더 이름입니다. 값을 포함하는 첫 번째 헤더가 ID로 사용됩니다. 필수 항목이며 대소문자를 구분하지 않습니다.

8 이메일 주소를 순서대로 확인할 헤더 이름입니다. 값을 포함하는 첫 번째 헤더가 이메일 주소로 사용됩니다. 선택 항목이며 대소문자를 구분하지 않습니다.

9 표시 이름을 순서대로 확인할 헤더 이름입니다. 값을 포함하는 첫 번째 헤더가 표시 이름으로 사용됩니다. 선택 항목이며 대소문자를 구분하지 않습니다.

10 **headers**에 지정된 헤더에서 결정된 불변 ID와 다른 경우, 기본 사용자 이름을 순서대로 확인하는 헤더 이름입니다. 값이 포 된 첫 번째 헤더는 프로비저닝시 기본 사용자 이름으로 사용됩니다. 선택 항목이며 대소문자를 구분하지 않습니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.5.5. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```

참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

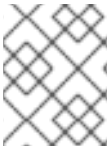
```
$ oc whoami
```

7.5.6. 요청 헤더를 사용하는 Apache 인증 구성 예

이 예제에서는 요청 헤더 ID 공급자를 사용하여 OpenShift Container Platform에 대한 Apache 인증 프록시를 구성합니다.

사용자 정의 프록시 구성
mod_auth_gssapi 모듈을 사용하는 것은 요청 헤더 ID 공급자를 사용하여 Apache 인증 프록시를 구성하는 일반적인 방법입니다. 그러나 필수는 아닙니다. 다음 요구 사항이 충족되면 다른 프록시를 쉽게 사용할 수 있습니다.

- 스푸핑을 방지하기 위해 클라이언트 요청에서 **X-Remote-User** 헤더를 차단합니다.
- **RequestHeaderIdentityProvider** 구성에서 클라이언트 인증서 인증을 시행합니다.
- 챌린지 flow를 사용하는 모든 인증 요청에 대해 **X-Csrf-Token** 헤더를 설정해야 합니다.
- **/oauth/authorize** 끝점 및 해당 하위 경로만 프록시되는지 확인하십시오. 백엔드 서버에서 클라이언트를 올바른 위치로 보낼 수 있도록 리디렉션을 다시 작성해야 합니다.
- **https://<namespace_route>/oauth/authorize**로 프록시되는 URL은 후행 슬래시 없이 **/authorize**로 끝나야 합니다. 예: **https://proxy.example.com/login-proxy/authorize?… https://<namespace_route>/oauth/authorize?…로 프록시해야 합니다.**.
- **https://<namespace_route>/oauth/authorize**로 프록시되는 URL 하위 경로는 **https://<namespace_route>/oauth/authorize** 하위 경로로 프록시되어야 합니다. 예: **https://proxy.example.com/login-proxy/authorize/approve?… https://<namespace_route>/oauth/authorize/approve?…로 프록시해야 합니다.**.



참고

https://<namespace_route> 주소는 OAuth 서버로의 경로이며 **oc get route -n openshift-authentication**을 실행하여 가져올 수 있습니다.

요청 헤더를 사용하여 Apache 인증 구성
이 예제에서는 **mod_auth_gssapi** 모듈을 사용하여 요청 헤더 ID 공급자를 통한 Apache 인증 프록시를 구성합니다.

사전 요구 사항

- 선택적 채널에서 **mod_auth_gssapi** 모듈을 가져옵니다. 로컬 시스템에 다음 패키지가 설치되어 있어야 합니다.
  - **httpd**
  - **mod_ssl**
  - **mod_session**
  - **apr-util-openssl**
  - **mod_auth_gssapi**
- 신뢰할 수 있는 헤더를 제출하는 요청을 검증하는 CA를 생성합니다. CA가 포함된 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 다음을 실행하여 수행합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config 1
```

1 CA는 **ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

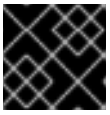
```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

- 프록시의 클라이언트 인증서를 생성합니다. 이 인증서는 x509 인증서 툴링을 사용하여 생성할 수 있습니다. 클라이언트 인증서는 신뢰할 수 있는 헤더를 제출하는 요청을 검증하기 위해 생성한 CA에서 서명해야 합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.

절차

이 프록시는 클라이언트 인증서를 사용하여 **X-Remote-User** 헤더를 신뢰하도록 구성된 OAuth 서버에 연결합니다.

1. Apache 구성에 대한 인증서를 생성합니다. **SSLProxyMachineCertificateFile** 매개변수 값으로 지정하는 인증서는 서버에 프록시를 인증하는 데 사용되는 프록시의 클라이언트 인증서입니다. 확장 키 유형으로 **TLS 웹 클라이언트 인증**을 사용해야 합니다.
2. Apache 구성을 생성합니다. 다음 템플릿을 사용하여 필요한 설정 및 값을 제공하십시오.



중요

템플릿을 신중하게 검토하고 환경에 맞게 내용을 사용자 정의하십시오.

```
LoadModule request_module modules/mod_request.so
LoadModule auth_gssapi_module modules/mod_auth_gssapi.so
# Some Apache configurations might require these modules.
# LoadModule auth_form_module modules/mod_auth_form.so
# LoadModule session_module modules/mod_session.so

# Nothing needs to be served over HTTP.  This virtual host simply redirects to
# HTTPS.
<VirtualHost *:80>
  DocumentRoot /var/www/html
  RewriteEngine              On
  RewriteRule     ^(.*)$     https://%{HTTP_HOST}$1 [R,L]
</VirtualHost>

<VirtualHost *:443>
  # This needs to match the certificates you generated.  See the CN and X509v3
  # Subject Alternative Name in the output of:
  # openssl x509 -text -in /etc/pki/tls/certs/localhost.crt
  ServerName www.example.com

  DocumentRoot /var/www/html
  SSLEngine on
  SSLCertificateFile /etc/pki/tls/certs/localhost.crt
  SSLCertificateKeyFile /etc/pki/tls/private/localhost.key
  SSLCACertificateFile /etc/pki/CA/certs/ca.crt

  SSLProxyEngine on
  SSLProxyCACertificateFile /etc/pki/CA/certs/ca.crt
  # It is critical to enforce client certificates. Otherwise, requests can
  # spoof the X-Remote-User header by accessing the /oauth/authorize endpoint
  # directly.
  SSLProxyMachineCertificateFile /etc/pki/tls/certs/authproxy.pem

  # To use the challenging-proxy, an X-Csrf-Token must be present.
  RewriteCond %{REQUEST_URI} ^/challenging-proxy
  RewriteCond %{HTTP:X-Csrf-Token} ^$ [NC]
  RewriteRule ^.* - [F,L]

  <Location /challenging-proxy/oauth/authorize>
     # Insert your backend server name/ip here.
     ProxyPass https://<namespace_route>/oauth/authorize
     AuthName "SSO Login"
     # For Kerberos
     AuthType GSSAPI
     Require valid-user
     RequestHeader set X-Remote-User %{REMOTE_USER}s

     GssapiCredStore keytab:/etc/httpd/protected/auth-proxy.keytab
     # Enable the following if you want to allow users to fallback
     # to password based authentication when they do not have a client
     # configured to perform kerberos authentication.
```

```
      GssapiBasicAuth On

      # For ldap:
      # AuthBasicProvider ldap
      # AuthLDAPURL "ldap://ldap.example.com:389/ou=People,dc=my-domain,dc=com?uid?
sub?(objectClass=*)"
    </Location>

    <Location /login-proxy/oauth/authorize>
    # Insert your backend server name/ip here.
    ProxyPass https://<namespace_route>/oauth/authorize

      AuthName "SSO Login"
      AuthType GSSAPI
      Require valid-user
      RequestHeader set X-Remote-User %{REMOTE_USER}s env=REMOTE_USER

      GssapiCredStore keytab:/etc/httpd/protected/auth-proxy.keytab
      # Enable the following if you want to allow users to fallback
      # to password based authentication when they do not have a client
      # configured to perform kerberos authentication.
      GssapiBasicAuth On

      ErrorDocument 401 /login.html
    </Location>

</VirtualHost>

RequestHeader unset X-Remote-User
```

참고

https://<namespace_route> 주소는 OAuth 서버로의 경로이며 **oc get route -n openshift-authentication**을 실행하여 가져올 수 있습니다.

3. CR(사용자 정의 리소스)에서 **identityProviders** 스탠자를 업데이트합니다.

```
identityProviders:
  - name: requestheaderidp
    type: RequestHeader
    requestHeader:
      challengeURL: "https://<namespace_route>/challenging-proxy/oauth/authorize?${query}"
      loginURL: "https://<namespace_route>/login-proxy/oauth/authorize?${query}"
      ca:
        name: ca-config-map
        clientCommonNames:
        - my-auth-proxy
        headers:
        - X-Remote-User
```

4. 구성을 확인합니다.

   a. 올바른 클라이언트 인증서 및 헤더를 제공하는 방식으로 토큰을 요청하여 프록시를 바이패스할 수 있는지 확인합니다.

```
# curl -L -k -H "X-Remote-User: joe" \
  --cert /etc/pki/tls/certs/authproxy.pem \
  https://<namespace_route>/oauth/token/request
```

b. 인증서 없이 토큰을 요청하여 클라이언트 인증서를 제공하지 않는 요청이 실패하는지 확인합니다.

```
# curl -L -k -H "X-Remote-User: joe" \
  https://<namespace_route>/oauth/token/request
```

c. **challengeURL** 리디렉션이 활성화되어 있는지 확인합니다.

```
# curl -k -v -H 'X-Csrf-Token: 1' \
  https://<namespace_route>/oauth/authorize?client_id=openshift-challenging-
client&response_type=token
```

다음 단계에서 사용할 **challengeURL** 리디렉션을 복사합니다.

d. **WWW-Authenticate** 기본 챌린지, 협상 챌린지 또는 두 가지 챌린지로 **401** 응답을 표시하려면 이 명령을 실행합니다.

```
# curl -k -v -H 'X-Csrf-Token: 1' \
  <challengeURL_redirect + query>
```

e. Kerberos 티켓을 사용하거나 사용하지 않고 OpenShift CLI(**oc**) 로그인을 테스트합니다.

i. **kinit**를 사용하여 Kerberos 티켓을 생성한 경우 이를 삭제합니다.

```
# kdestroy -c cache_name 1
```

1 Kerberos 캐시의 이름을 제공해야 합니다.

ii. Kerberos 인증 정보를 사용하여 **oc** 도구에 로그인합니다.

```
# oc login -u <username>
```

프롬프트에 Kerberos 암호를 입력합니다.

iii. **oc** 도구에서 로그아웃합니다.

```
# oc logout
```

iv. Kerberos 인증 정보를 사용하여 티켓을 받습니다.

```
# kinit
```

프롬프트에 Kerberos 사용자 이름과 암호를 입력합니다.

v. **oc** 도구에 로그인할 수 있는지 확인합니다.

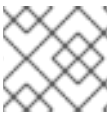
```
# oc login
```

구성이 올바르면 별도의 인증 정보를 입력하지 않아도 로그인할 수 있습니다.

7.6. GITHUB 또는 GITHUB ENTERPRISE ID 공급자 구성

GitHub 또는 GitHub Enterprise의 OAuth 인증 서버에 대해 사용자 이름 및 암호의 유효성을 확인하도록 **github** ID 공급자를 구성합니다. OAuth를 사용하면 OpenShift Container Platform과 GitHub 또는 GitHub Enterprise 간의 토큰 교환 flow가 용이해집니다.

GitHub 통합을 사용하여 GitHub 또는 GitHub Enterprise에 연결할 수 있습니다. GitHub Enterprise 통합의 경우 인스턴스의 **호스트 이름**을 제공해야 하며, 서버에 대한 요청에 사용할 **ca** 인증서 번들을 선택적으로 제공할 수 있습니다.



참고

다음 단계는 별도로 명시하지 않는 한 GitHub 및 GitHub Enterprise에 모두 적용됩니다.

7.6.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.6.2. GitHub 인증 정보

GitHub 인증을 구성하면 사용자가 GitHub 자격 증명을 사용하여 OpenShift Container Platform에 로그인할 수 있습니다. GitHub 사용자 ID가 있는 사람이 OpenShift Container Platform 클러스터에 로그인하지 못하도록 특정 GitHub 조직의 사용자만 액세스할 수 있도록 제한할 수 있습니다.

7.6.3. GitHub 애플리케이션 등록

GitHub 또는 GitHub Enterprise를 ID 공급자로 사용하려면 사용할 애플리케이션을 등록해야 합니다.

절차

1. GitHub에 애플리케이션을 등록합니다.
   - GitHub의 경우 **Settings → Developer settings → OAuth Apps → Register a new OAuth application**을 클릭합니다.
   - GitHub Enterprise의 경우 GitHub Enterprise 홈페이지로 이동한 다음 **Settings → Developer settings → Register a new application**을 클릭합니다.
2. 애플리케이션 이름(예: **My OpenShift Install)**을 입력합니다.
3. **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>**과 같은 홈페이지 URL을 입력합니다.
4. 선택 사항: 애플리케이션 설명을 입력합니다.
5. 권한 부여 콜백 URL을 입력합니다. URL 끝에는 ID 공급자 **name**이 있습니다.

```
https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>
```

예를 들면 다음과 같습니다.

```
https://oauth-openshift.apps.openshift-cluster.example.com/oauth2callback/github
```

6. **Register application**을 클릭합니다. GitHub에서 클라이언트 ID와 클라이언트 시크릿을 제공합니다. ID 공급자 구성을 완료하려면 이러한 값이 필요합니다.

7.6.4. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 문자열이 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic <secret_name> --from-literal=clientSecret=<secret> -n openshift-config
```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 다음 명령을 사용하여 인증서 파일과 같은 파일 내용이 포함된 **Secret** 오브젝트를 정의할 수 있습니다.

```
$ oc create secret generic <secret_name> --from-file=<path_to_file> -n openshift-config
```

7.6.5. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

참고

이 절차는 GitHub Enterprise에만 필요합니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.6.6. GitHub CR 샘플

다음 CR(사용자 정의 리소스)에는 GitHub ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

GitHub CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: githubidp (1)
    mappingMethod: claim (2)
    type: GitHub
    github:
      ca: (3)
        name: ca-config-map
      clientID: {...} (4)
      clientSecret: (5)
        name: github-secret
      hostname: ... (6)
      organizations: (7)
      - myorganization1
      - myorganization2
      teams: (8)
      - myorganization1/team-a
      - myorganization2/team-b
```

(1) 이 공급자 이름은 GitHub 숫자 사용자 ID 앞에 접두어로 지정되어 ID 이름을 형성합니다. 콜백 URL을 빌드하는 데에도 사용됩니다.

(2) 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

3 선택 사항: OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다. 공개적으로

4 등록된 GitHub OAuth 애플리케이션의 클라이언트 ID. 애플리케이션은 **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**의 콜백 URL을 사용하여 구성해야 합니다.

5 OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, GitHub에서 발행한 클라이언트 시크릿이 포함됩니다.

6 GitHub Enterprise의 경우 인스턴스의 호스트 이름(예 **: example.com**)을 제공해야 합니다. 이 값은 **/setup/settings** 파일의 GitHub Enterprise **hostname** 값과 일치해야 하며 포트 번호를 포함할 수 없습니다. 이 값을 설정하지 않으면 **teams** 또는 **organizations**을 정의해야 합니다. GitHub의 경우 이 매개변수를 생략합니다.

7 조직 목록. **hostname** 필드가 설정되어 있지 않거나 **mappingMethod**가 **lookup**으로 설정되어 있는 경우에는 **organizations** 또는 **teams** 필드를 설정해야 합니다. **teams** 필드와 함께 사용할 수 없습니다.

8 팀 목록. **hostname** 필드가 설정되어 있지 않거나 **mappingMethod**가 **lookup**으로 설정되어 있는 경우에는 **teams** 또는 **organizations** 필드를 설정해야 합니다. **organizations** 필드와 함께 사용할 수 없습니다.

참고

organizations 또는 **teams**가 지정된 경우 나열된 조직 중 하나 이상에 속하는 GitHub 사용자만 로그인할 수 있습니다. **clientID**에 구성된 GitHub OAuth 애플리케이션이 조직의 소유가 아닌 경우 조직 소유자가 이 옵션을 사용하려면 타사 액세스 권한을 부여해야 합니다. 이러한 작업은 조직 관리자가 GitHub를 처음 로그인하는 동안 또는 GitHub 조직 설정에서 수행할 수 있습니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.6.7. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```

참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. OAuth 서버에서 토큰을 가져옵니다.
kubeadmin 사용자가 제거된 경우 **oc login** 명령을 실행하면 토큰을 검색할 수 있는 웹 페이지에 액세스하는 방법에 대한 지침이 제공됩니다.

   웹 콘솔에서 **(?) Help → Command Line Tools→ Copy Login Command**로 이동하여 이 페이지에 액세스할 수도 있습니다.

3. 클러스터에 로그인하여 인증을 위해 토큰을 전달합니다.

```
$ oc login --token=<token>
```

참고

이 ID 공급자는 사용자 이름과 암호를 사용한 로그인을 지원하지 않습니다.

4. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

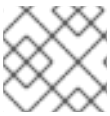
```
$ oc whoami
```

7.7. GITLAB ID 공급자 구성

GitLab.com 또는 기타 GitLab 인스턴스를 ID 공급자로 사용하여 **gitlab** ID 공급자를 구성합니다.

7.7.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.



참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.7.2. GitLab 인증 정보

GitLab 인증을 구성하면 사용자가 GitLab 인증 정보를 사용하여 OpenShift Container Platform에 로그인할 수 있습니다.

GitLab 버전 7.7.0~11.0을 사용하는 경우 OAuth 통합을 사용하여 연결합니다. GitLab 버전 11.1 이상을 사용하는 경우 OAuth 대신 OpenID Connect(OIDC)를 사용하여 연결할 수 있습니다.

7.7.3. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 문자열이 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic <secret_name> --from-literal=clientSecret=<secret> -n openshift-config
```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 다음 명령을 사용하여 인증서 파일과 같은 파일 내용이 포함된 **Secret** 오브젝트를 정의할 수 있습니다.

```
$ oc create secret generic <secret_name> --from-file=<path_to_file> -n openshift-config
```

7.7.4. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

참고

이 절차는 GitHub Enterprise에만 필요합니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.7.5. GitLab CR 샘플

다음 CR(사용자 정의 리소스)에는 GitLab ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

GitLab CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: gitlabidp 1
    mappingMethod: claim 2
    type: GitLab
    gitlab:
      clientID: {...} 3
      clientSecret: 4
        name: gitlab-secret
      url: https://gitlab.com 5
      ca: 6
        name: ca-config-map
```

1. 이 공급자 이름은 GitLab 숫자 사용자 ID 앞에 접두어로 지정되어 ID 이름을 형성합니다. 콜백 URL을 빌드하는 데에도 사용됩니다.
2. 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.
3. 등록된 GitLab OAuth 애플리케이션의 클라이언트 ID. 애플리케이션은 **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**의 콜백 URL을 사용하여 구성해야 합니다.
4. OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, GitLab에서 발행한 클라이언트 시크릿이 포함됩니다.
5. GitLab 공급자의 호스트 URL. **https://gitlab.com/** 또는 기타 자체 호스팅 GitLab 인스턴스일 수 있습니다.
6. 선택 사항: OpenShift Container Platform **ConfigMap** 오브젝트에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.7.6. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

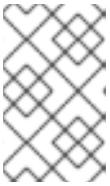
사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```



참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. 암호를 입력하라는 메시지가 표시되면 암호를 입력하여 ID 공급자의 사용자로 클러스터에 로그인합니다.

```
$ oc login -u <username>
```

3. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

```
$ oc whoami
```

7.8. GOOGLE ID 공급자 구성

Google OpenID Connect 통합을 사용하여 **google** ID 공급자를 구성합니다.

7.8.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.

참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.8.2. Google 인증 정보

Google을 ID 공급자로 사용하면 모든 Google 사용자가 서버 인증을 수행할 수 있습니다. **hostedDomain** 구성 속성을 사용하여 특정 호스트 도메인의 멤버 인증을 제한할 수 있습니다.

참고

Google을 ID 공급자로 사용하려면 사용자가 < **namespace_route>/oauth/token/request** 를 사용하여 토큰을 가져와야 합니다.

7.8.3. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 문자열이 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic <secret_name> --from-literal=clientSecret=<secret> -n openshift-config
```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 다음 명령을 사용하여 인증서 파일과 같은 파일 내용이 포함된 **Secret** 오브젝트를 정의할 수 있습니다.

```
$ oc create secret generic <secret_name> --from-file=<path_to_file> -n openshift-config
```

7.8.4. Google CR 샘플

다음 CR(사용자 정의 리소스)에는 Google ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

Google CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: googleidp 1
```

```
  mappingMethod: claim (2)
  type: Google
  google:
    clientID: {...} (3)
    clientSecret: (4)
      name: google-secret
    hostedDomain: "example.com" (5)
```

(1) 이 공급자 이름은 Google 숫자 사용자 ID 앞에 접두어로 지정되어 ID 이름을 형성합니다. 리디렉션 URL을 빌드하는 데에도 사용됩니다.

(2) 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.

(3) 등록된 Google 프로젝트의 클라이언트 ID. 프로젝트는 **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**의 리디렉션 URI를 사용하여 구성해야 합니다.

(4) OpenShift Container Platform **Secret** 오브젝트에 대한 참조로, Google에서 발행한 클라이언트 시크릿이 포함됩니다.

(5) 로그인 계정을 제한하는 데 사용되는 호스트 도메인입니다. **lookup mappingMethod**가 사용되는 경우 선택 사항입니다. 비어있는 경우 모든 Google 계정을 인증할 수 있습니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.8.5. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

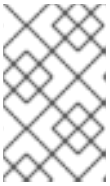
사전 요구 사항

- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.
- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```



참고

CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. OAuth 서버에서 토큰을 가져옵니다.

kubeadmin 사용자가 제거된 경우 **oc login** 명령을 실행하면 토큰을 검색할 수 있는 웹 페이지에 액세스하는 방법에 대한 지침이 제공됩니다.

웹 콘솔에서 **(?) Help → Command Line Tools→ Copy Login Command**로 이동하여 이 페이지에 액세스할 수도 있습니다.

3. 클러스터에 로그인하여 인증을 위해 토큰을 전달합니다.

```
$ oc login --token=<token>
```

참고

이 ID 공급자는 사용자 이름과 암호를 사용한 로그인을 지원하지 않습니다.

4. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

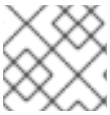
```
$ oc whoami
```

7.9. OPENID CONNECT ID 공급자 구성

인증 코드 Flow 를 사용하여 OpenID Connect ID 공급자와 통합하도록 **oidc** ID 공급자를 구성합니다.

7.9.1. OpenShift Container Platform의 ID 공급자 정보

기본적으로는 **kubeadmin** 사용자만 클러스터에 있습니다. ID 공급자를 지정하려면 해당 ID 공급자를 설명하는 CR(사용자 정의 리소스)을 생성하여 클러스터에 추가해야 합니다.



참고

/, **:**, **%**를 포함하는 OpenShift Container Platform 사용자 이름은 지원되지 않습니다.

7.9.2. 시크릿 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **Secret** 오브젝트를 사용하여 클라이언트 시크릿, 클라이언트 인증서 및 키를 포함합니다.

절차

- 다음 명령을 사용하여 문자열이 포함된 **Secret** 오브젝트를 생성합니다.

```
$ oc create secret generic <secret_name> --from-literal=clientSecret=<secret> -n openshift-config
```

작은 정보

다음 YAML을 적용하여 시크릿을 생성할 수도 있습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 다음 명령을 사용하여 인증서 파일과 같은 파일 내용이 포함된 **Secret** 오브젝트를 정의할 수 있습니다.

```
$ oc create secret generic <secret_name> --from-file=<path_to_file> -n openshift-config
```

7.9.3. 구성 맵 생성

ID 공급자는 **openshift-config** 네임스페이스에서 OpenShift Container Platform **ConfigMap** 오브젝트를 사용하여 인증 기관 번들을 포함합니다. 이들은 주로 ID 공급자에 필요한 인증서 번들을 포함하는 데 사용됩니다.

참고

이 절차는 GitHub Enterprise에만 필요합니다.

절차

- 다음 명령을 사용하여 인증 기관을 포함하는 OpenShift Container Platform **ConfigMap** 오브젝트를 정의합니다. 인증 기관은**ConfigMap** 오브젝트의 **ca.crt** 키에 저장해야 합니다.

```
$ oc create configmap ca-config-map --from-file=ca.crt=/path/to/ca -n openshift-config
```

작은 정보

다음 YAML을 적용하여 구성 맵을 만들 수 있습니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ca-config-map
  namespace: openshift-config
data:
  ca.crt: |
    <CA_certificate_PEM>
```

7.9.4. OpenID Connect CR 샘플

다음 CR(사용자 정의 리소스)에는 OpenID Connect ID 공급자에 대한 매개변수 및 허용 가능한 값이 표시되어 있습니다.

사용자 정의 인증서 번들, 추가 범위, 추가 권한 부여 요청 매개변수 또는 **userInfo** URL을 지정해야 하는 경우 전체 OpenID Connect CR을 사용하십시오.

표준 OpenID Connect CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: oidcidp 1
    mappingMethod: claim 2
    type: OpenID
    openID:
      clientID: ... 3
      clientSecret: 4
        name: idp-secret
      claims: 5
        preferredUsername:
        - preferred_username
        name:
        - name
        email:
        - email
      issuer: https://www.idp-issuer.com 6
```

1. 이 공급자 이름은 ID 클레임 값 앞에 접두어로 지정되어 ID 이름을 형성합니다. 리디렉션 URL을 빌드하는 데에도 사용됩니다.
2. 이 공급자의 ID와 **User** 오브젝트 간 매핑 설정 방법을 제어합니다.
3. OpenID 공급자에 등록된 클라이언트의 클라이언트 ID. 클라이언트를 **https://oauth-openshift.apps.<cluster_name>.<cluster_domain>/oauth2callback/<idp_provider_name>**으로 리디렉션할 수 있어야 합니다.
4. 클라이언트 시크릿이 포함된 OpenShift Container Platform **Secret** 오브젝트에 대한 참조입니다.
5. ID로 사용할 클레임 목록입니다. 비어 있지 않은 첫 번째 클레임이 사용됩니다. 클레임이 한 개 이상 있어야 합니다. 나열된 클레임에 값이 없으면 인증이 실패합니다. 예를 들어, 여기서는 반환된 **id_token**의 **sub** 클레임 값을 사용자의 ID로 사용합니다.
6. OpenID 사양에 설명된 발행자 식별자입니다. 쿼리 또는 조각 구성 요소 없이 **https**를 사용해야 합니다.

전체 OpenID Connect CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: oidcidp
```

```
    mappingMethod: claim
    type: OpenID
    openID:
      clientID: ...
      clientSecret:
        name: idp-secret
      ca: (1)
        name: ca-config-map
      extraScopes: (2)
      - email
      - profile
      extraAuthorizeParameters: (3)
        include_granted_scopes: "true"
      claims:
        preferredUsername: (4)
        - preferred_username
        - email
        name: (5)
        - nickname
        - given_name
        - name
        email: (6)
        - custom_email_claim
        - email
      issuer: https://www.idp-issuer.com
```

(1) 선택 사항: OpenShift Container Platform 구성 맵에 대한 참조로, 구성된 URL에 대한 서버 인증서의 유효성을 검증하는 데 사용할 PEM 인코딩 인증 기관 번들이 포함됩니다.

(2) 권한 부여 토큰 요청 중 **openid** 범위 외에 요청할 선택적 범위 목록입니다.

(3) 권한 부여 토큰 요청에 추가할 추가 매개변수의 선택적 맵입니다.

(4) 이 ID에 대해 사용자를 프로비저닝할 때 기본 사용자 이름으로 사용할 클레임 목록입니다. 비어 있지 않은 첫 번째 클레임이 사용됩니다.

(5) 표시 이름으로 사용할 클레임 목록입니다. 비어 있지 않은 첫 번째 클레임이 사용됩니다.

(6) 이메일 주소로 사용할 클레임 목록입니다. 비어 있지 않은 첫 번째 클레임이 사용됩니다.

추가 리소스

- 모든 ID 공급자에 공통되는 **mappingMethod**와 같은 매개변수에 대한 자세한 내용은 ID 공급자 매개변수를 참조하십시오.

7.9.5. 클러스터에 ID 공급자 추가

클러스터를 설치한 후에는 사용자가 인증할 수 있도록 ID 공급자를 추가하십시오.

사전 요구 사항

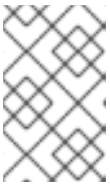
- OpenShift Container Platform 클러스터를 생성합니다.
- ID 공급자의 CR(사용자 정의 리소스)을 만듭니다.

- 관리자로 로그인해야 합니다.

절차

1. 정의된 CR을 적용합니다.

```
$ oc apply -f </path/to/CR>
```



참고

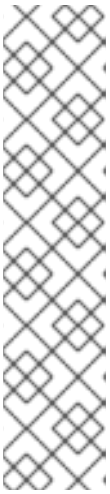
CR이 없으면 **oc apply** 에서 새 CR을 생성하고 다음 경고를 트리거할 수 있습니다. **경고: oc apply는 oc create --save-config 또는 oc apply에서 생성한 리소스에 사용해야 합니다**. 이 경우 이 경고를 무시해도 됩니다.

2. OAuth 서버에서 토큰을 가져옵니다.
kubeadmin 사용자가 제거된 경우 **oc login** 명령을 실행하면 토큰을 검색할 수 있는 웹 페이지에 액세스하는 방법에 대한 지침이 제공됩니다.

   웹 콘솔에서 **(?) Help → Command Line Tools→ Copy Login Command**로 이동하여 이 페이지에 액세스할 수도 있습니다.

3. 클러스터에 로그인하여 인증을 위해 토큰을 전달합니다.

```
$ oc login --token=<token>
```



참고

OpenID Connect 아이덴티티 공급자가 ROPC(Resource Owner Password Credentials) 허가 흐름을 지원하는 경우 사용자 이름과 암호로 로그인할 수 있습니다. 아이덴티티 공급자에 대한 ROPC 허가 흐름을 사용하도록 활성화하는 단계를 수행해야 할 수 있습니다.

OIDC 아이덴티티 공급자가 OpenShift Container Platform에 구성된 후 다음 명령을 사용하여 로그인할 수 있습니다. 이 경우 사용자 이름과 암호를 입력하라는 메시지가 표시됩니다.

```
$ oc login -u <identity_provider_username> --server=
<api_server_url_and_port>
```

4. 사용자가 로그인했는지 확인하고 사용자 이름을 표시합니다.

```
$ oc whoami
```

7.9.6. 웹 콘솔을 사용하여 ID 공급자 구성

CLI 대신 웹 콘솔을 통해 ID 공급자(IDP)를 구성합니다.

사전 요구 사항

- 웹 콘솔에 클러스터 관리자로 로그인해야 합니다.

절차

1. **관리 → 클러스터 설정**으로 이동합니다.
2. **글로벌 구성** 탭에서 **OAuth**를 클릭합니다.
3. **ID 공급자** 섹션의 **추가** 드롭다운 메뉴에서 ID 공급자를 선택합니다.

참고

기존 IDP를 덮어쓰지 않고 웹 콘솔을 통해 여러 IDP를 지정할 수 있습니다.

8장. RBAC를 사용하여 권한 정의 및 적용

8.1. RBAC 개요

RBAC(역할 기반 액세스 제어) 오브젝트에 따라 사용자가 프로젝트 내에서 지정된 작업을 수행할 수 있는지가 결정됩니다.

클러스터 관리자는 클러스터 역할 및 바인딩을 사용하여 OpenShift Container Platform 플랫폼 자체 및 모든 프로젝트에 대해 다양한 액세스 수준을 보유한 사용자를 제어할 수 있습니다.

개발자는 로컬 역할 및 바인딩을 사용하여 프로젝트에 액세스할 수 있는 사용자를 제어할 수 있습니다. 권한 부여는 인증과 별도의 단계이며, 여기서는 조치를 수행할 사용자의 신원을 파악하는 것이 더 중요합니다.

권한 부여는 다음을 사용하여 관리합니다.

권한 부여 오브젝트	설명
규칙	오브젝트 집합에 허용되는 동사 집합입니다. 예를 들면 사용자 또는 서비스 계정의 Pod **생성** 가능 여부입니다.
역할	규칙 모음입니다. 사용자와 그룹을 여러 역할에 연결하거나 바인딩할 수 있습니다.
바인딩	역할이 있는 사용자 및/또는 그룹 간 연결입니다.

권한 부여를 제어하는 두 가지 수준의 RBAC 역할 및 바인딩이 있습니다.

RBAC 수준	설명
클러스터 RBAC	모든 프로젝트에 적용할 수 있는 역할 및 바인딩입니다. *클러스터 역할*은 클러스터 전체에 존재하며 *클러스터 역할 바인딩*은 클러스터 역할만 참조할 수 있습니다.
지역 RBAC	지정된 프로젝트에 적용되는 역할 및 바인딩입니다. *로컬 역할*은 단일 프로젝트에만 존재하지만 로컬 역할 바인딩은 클러스터 및 로컬 역할을 *모두* 참조할 수 있습니다.

클러스터 역할 바인딩은 클러스터 수준에 존재하는 바인딩입니다. 역할 바인딩은 프로젝트 수준에 있습니다. 해당 사용자가 프로젝트를 보려면 로컬 역할 바인딩을 사용하여 클러스터 역할 *보기*를 사용자에게 바인딩해야 합니다. 클러스터 역할이 특정 상황에 필요한 권한 집합을 제공하지 않는 경우에만 로컬 역할을 생성하십시오.

이러한 2단계 계층 구조로 인해 클러스터 역할로는 여러 프로젝트에서 재사용하고, 로컬 역할로는 개별 프로젝트 내에서 사용자 정의할 수 있습니다.

평가 중에는 클러스터 역할 바인딩과 로컬 역할 바인딩이 모두 사용됩니다. 예를 들면 다음과 같습니다.

1. 클러스터 전체의 "허용" 규칙을 확인합니다.
2. 로컬 바인딩된 "허용" 규칙을 확인합니다.
3. 기본적으로 거부합니다.

8.1.1. 기본 클러스터 역할

OpenShift Container Platform에는 클러스터 전체 또는 로컬로 사용자 및 그룹에 바인딩할 수 있는 기본 클러스터 역할 집합이 포함되어 있습니다.

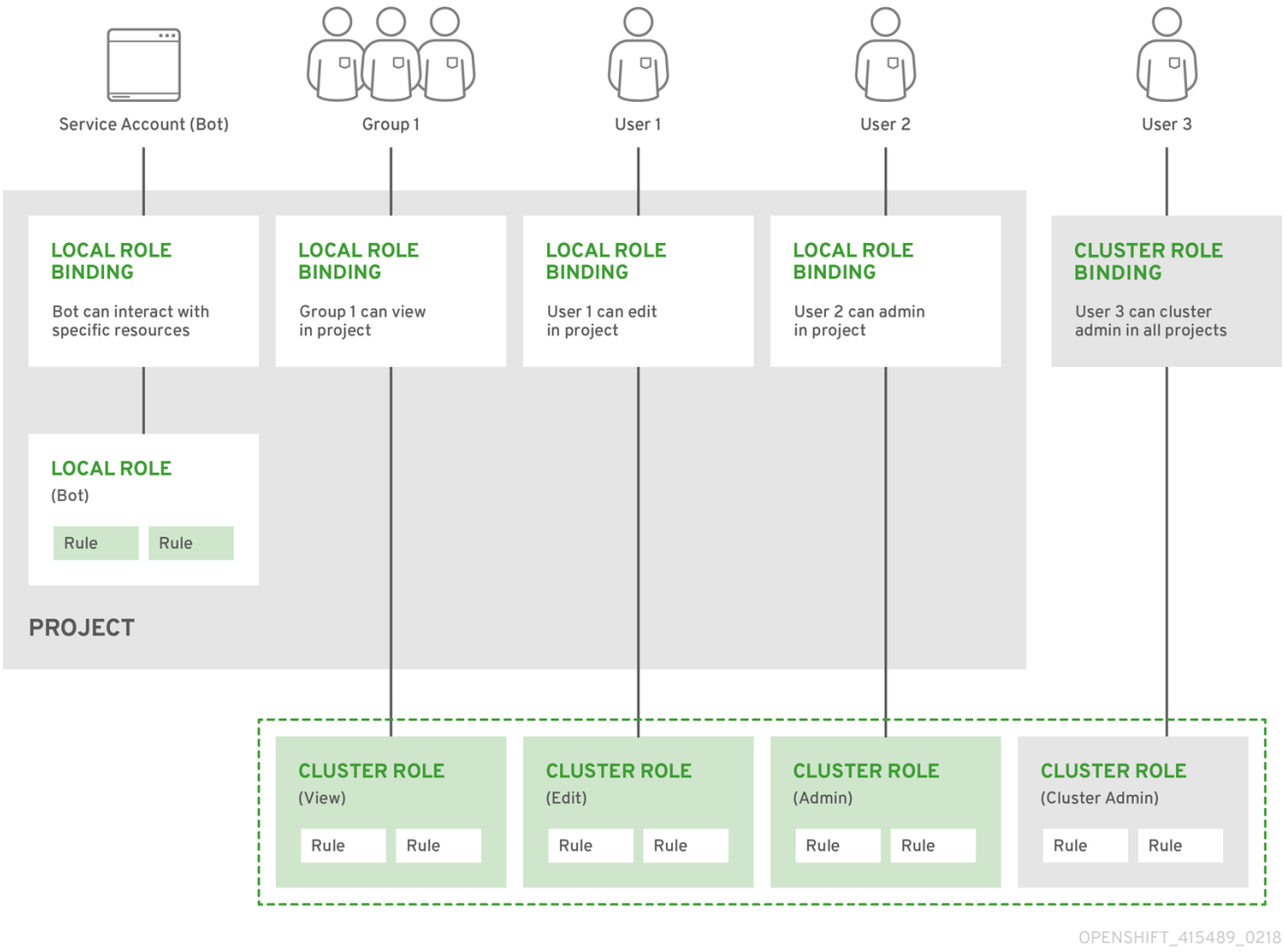
중요

기본 클러스터 역할을 수동으로 수정하지 않는 것이 좋습니다. 이러한 시스템 역할에 대한 수정으로 인해 클러스터가 제대로 작동하지 않을 수 있습니다.

기본 클러스터 역할	설명
admin	프로젝트 관리자입니다. 로컬 바인딩에 사용되는 경우 **admin**은 프로젝트의 모든 리소스를 보고 할당량을 제외한 프로젝트의 모든 리소스를 수정할 수 있는 권한이 있습니다.
basic-user	프로젝트 및 사용자에 대한 기본 정보를 가져올 수 있는 사용자입니다.
cluster-admin	모든 프로젝트에서 모든 작업을 수행할 수 있는 슈퍼 유저입니다. 로컬 바인딩을 통해 사용자에게 바인딩하면 할당량은 물론 프로젝트의 모든 리소스에 대한 모든 조치를 완전히 제어할 수 있습니다.
cluster-status	기본 클러스터 상태 정보를 가져올 수 있는 사용자입니다.
cluster-reader	대부분의 개체를 가져오거나 볼 수 있지만 수정할 수는 없는 사용자입니다.
edit	프로젝트에서 대부분의 오브젝트를 수정할 수 있지만 역할이나 바인딩을 보거나 수정할 권한은 없는 사용자입니다.
self-provisioner	자체 프로젝트를 만들 수 있는 사용자입니다.
view	수정할 수는 없지만 프로젝트의 오브젝트를 대부분 볼 수 있는 사용자입니다. 역할 또는 바인딩을 보거나 수정할 수 없습니다.

로컬 바인딩과 클러스터 바인딩의 차이점에 유의하십시오. 예를 들어 로컬 역할 바인딩을 사용하여 **cluster-admin** 역할을 사용자에게 바인딩하는 경우, 이 사용자에게 클러스터 관리자 권한이 있는 것처럼 보일 수 있습니다. 사실은 그렇지 않습니다. 프로젝트의 사용자에게 **cluster-admin**을 바인딩하면 해당 프로젝트에 대해서만 슈퍼 관리자 권한이 사용자에게 부여됩니다. 해당 사용자에게는 클러스터 역할 **admin**의 권한을 비롯하여 해당 프로젝트에 대한 속도 제한 편집 기능과 같은 몇 가지 추가 권한이 있습니다. 이 바인딩은 실제 클러스터 관리자에게 바인딩된 클러스터 역할 바인딩이 나열되지 않는 웹 콘솔 UI로 인해 혼동될 수 있습니다. 그러나 **cluster-admin**을 로컬로 바인딩하는 데 사용할 수 있는 로컬 역할 바인딩은 나열됩니다.

아래에는 클러스터 역할, 로컬 역할, 클러스터 역할 바인딩, 로컬 역할 바인딩, 사용자, 그룹, 서비스 계정 간의 관계가 설명되어 있습니다.



8.1.2. 권한 부여 평가

OpenShift Container Platform에서는 다음을 사용하여 권한 부여를 평가합니다.

ID

사용자 이름 및 사용자가 속한 그룹 목록입니다.

작업

수행하는 작업입니다. 대부분의 경우 다음으로 구성됩니다.

- **프로젝트**: 액세스하는 프로젝트입니다. 프로젝트는 추가 주석이 있는 쿠버네티스 네임스페이스로, 사용자 커뮤니티가 다른 커뮤니티와 별도로 컨텐츠를 구성하고 관리할 수 있습니다.
- **동사** : 작업 자체: **get**,**list**,**create**,**update**,**delete**,**deletecollection** 또는 **watch**.
- **리소스 이름**: 액세스하는 API 끝점입니다.

바인딩

전체 바인딩 목록으로, 역할이 있는 사용자 또는 그룹 간 연결을 나타냅니다.

OpenShift Container Platform에서는 다음 단계를 사용하여 권한 부여를 평가합니다.

1. ID 및 프로젝트 범위 작업은 사용자 또는 해당 그룹에 적용되는 모든 바인딩을 찾는 데 사용됩니다.
2. 바인딩은 적용되는 모든 역할을 찾는 데 사용됩니다.
3. 역할은 적용되는 모든 규칙을 찾는 데 사용됩니다.

4. 일치하는 규칙을 찾기 위해 작업을 각 규칙에 대해 확인합니다.

5. 일치하는 규칙이 없으면 기본적으로 작업이 거부됩니다.

작은 정보

사용자 및 그룹을 동시에 여러 역할과 연결하거나 바인딩할 수 있습니다.

프로젝트 관리자는 CLI를 사용하여 각각 연결된 동사 및 리소스 목록을 포함하여 로컬 역할 및 바인딩을 볼 수 있습니다.

중요

프로젝트 관리자에게 바인딩된 클러스터 역할은 로컬 바인딩을 통해 프로젝트에서 제한됩니다. **cluster-admin** 또는 **system:admin**에 부여되는 클러스터 역할과 같이 클러스터 전체에 바인딩되지 않습니다.

클러스터 역할은 클러스터 수준에서 정의된 역할이지만 클러스터 수준 또는 프로젝트 수준에서 바인딩할 수 있습니다.

8.1.2.1. 클러스터 역할 집계

기본 관리, 편집, 보기 및 클러스터 독자 클러스터 역할에서는 새 역할이 생성될 때 각 역할에 대한 클러스터 규칙이 동적으로 업데이트되는 클러스터 역할 집계를 지원합니다. 이 기능은 사용자 정의 리소스를 생성하여 쿠버네티스 API를 확장한 경우에만 관련이 있습니다.

8.2. 프로젝트 및 네임스페이스

쿠버네티스 *네임스페이스*는 클러스터의 리소스 범위를 지정하는 메커니즘을 제공합니다. 쿠버네티스 설명서에 네임스페이스에 대한 자세한 정보가 있습니다.

네임스페이스는 다음에 대한 고유 범위를 제공합니다.

- 기본 이름 지정 충돌을 피하기 위해 이름이 지정된 리소스
- 신뢰할 수 있는 사용자에게 위임된 관리 권한
- 커뮤니티 리소스 사용을 제한하는 기능

시스템에 있는 대부분의 오브젝트는 네임스페이스에 따라 범위가 지정되지만, 노드 및 사용자를 비롯한 일부는 여기에 해당하지 않으며 네임스페이스가 없습니다.

*프로젝트*는 추가 주석이 있는 쿠버네티스 네임스페이스이며, 일반 사용자용 리소스에 대한 액세스를 관리하는 가장 중요한 수단입니다. 사용자 커뮤니티는 프로젝트를 통해 다른 커뮤니티와 별도로 콘텐츠를 구성하고 관리할 수 있습니다. 사용자는 관리자로부터 프로젝트에 대한 액세스 권한을 부여받아야 합니다. 프로젝트를 생성하도록 허용된 경우 자신의 프로젝트에 액세스할 수 있는 권한이 자동으로 제공됩니다.

프로젝트에는 별도의 **name**, **displayName**, **description**이 있을 수 있습니다.

- 필수 항목인 **name**은 프로젝트의 고유 식별자이며 CLI 도구 또는 API를 사용할 때 가장 잘 보입니다. 최대 이름 길이는 63자입니다.
- 선택적 **displayName**은 프로젝트가 웹 콘솔에 표시되는 방법입니다(기본값: **name**).
- 선택적 **description**은 프로젝트에 대한 보다 자세한 설명으로, 웹 콘솔에서도 볼 수 있습니다.

각 프로젝트의 범위는 다음과 같습니다.

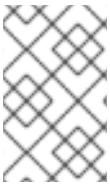
오브젝트	설명
Objects	Pod, 서비스, 복제 컨트롤러 등입니다.
Policies	사용자는 오브젝트에서 이 규칙에 대해 작업을 수행할 수 있거나 수행할 수 없습니다.
Constraints	제한할 수 있는 각 종류의 오브젝트에 대한 할당량입니다.
Service accounts	서비스 계정은 프로젝트의 오브젝트에 지정된 액세스 권한으로 자동으로 작동합니다.

클러스터 관리자는 프로젝트를 생성하고 프로젝트에 대한 관리 권한을 사용자 커뮤니티의 모든 멤버에게 위임할 수 있습니다. 클러스터 관리자는 개발자가 자신의 프로젝트를 만들 수 있도록 허용할 수도 있습니다.

개발자와 관리자는 CLI 또는 웹 콘솔을 사용하여 프로젝트와 상호 작용할 수 있습니다.

8.3. 기본 프로젝트

OpenShift Container Platform에는 다양한 기본 프로젝트가 제공되며, **openshift-**로 시작하는 프로젝트가 사용자에게 가장 중요합니다. 이러한 프로젝트는 Pod 및 기타 인프라 구성 요소로 실행되는 마스터 구성 요소를 호스팅합니다. 중요 Pod 주석 이 있는 네임스페이스에 생성된 Pod는 중요한 Pod로 간주되며, kubelet의 승인이 보장됩니다. 이러한 네임스페이스에서 마스터 구성 요소용으로 생성된 Pod는 이미 중요로 표시되어 있습니다.



참고

기본 네임스페이스(**default**, **kube-system**, **kube-public**, **openshift-node**, **openshift-infra** 및 **openshift**) 중 하나에서 생성된 Pod에는 SCC를 할당할 수 없습니다. 이러한 네임스페이스는 Pod 또는 서비스를 실행하는 데 사용할 수 없습니다.

8.4. 클러스터 역할 및 바인딩 보기

oc CLI에서 **oc describe** 명령을 사용하여 클러스터 역할 및 바인딩을 볼 수 있습니다.

사전 요구 사항

- **oc** CLI를 설치합니다.
- 클러스터 역할 및 바인딩을 볼 수 있는 권한을 얻습니다.

cluster-admin 기본 클러스터 역할이 클러스터 전체에서 바인딩된 사용자는 클러스터 역할 및 바인딩 보기를 포함하여 모든 리소스에 대해 모든 작업을 수행할 수 있습니다.

절차

1. 클러스터 역할 및 관련 규칙 집합을 보려면 다음을 수행합니다.

```
$ oc describe clusterrole.rbac
```

출력 예

```
Name:         admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources                                  Non-Resource URLs  Resource Names  Verbs
  ---------                                  -----------------  --------------  -----
  .packages.apps.redhat.com                  []                 []              [* create update
patch delete get list watch]
  imagestreams                               []                 []              [create delete
deletecollection get list patch update watch create get list watch]
  imagestreams.image.openshift.io            []                 []              [create delete
deletecollection get list patch update watch create get list watch]
  secrets                                    []                 []              [create delete deletecollection
get list patch update watch get list watch create delete deletecollection patch update]
  buildconfigs/webhooks                      []                 []              [create delete
deletecollection get list patch update watch get list watch]
  buildconfigs                               []                 []              [create delete
deletecollection get list patch update watch get list watch]
  buildlogs                                  []                 []              [create delete deletecollection
get list patch update watch get list watch]
  deploymentconfigs/scale                    []                 []              [create delete
deletecollection get list patch update watch get list watch]
  deploymentconfigs                          []                 []              [create delete
deletecollection get list patch update watch get list watch]
  imagestreamimages                          []                 []              [create delete
deletecollection get list patch update watch get list watch]
  imagestreammappings                        []                 []              [create delete
deletecollection get list patch update watch get list watch]
  imagestreamtags                            []                 []              [create delete
deletecollection get list patch update watch get list watch]
  processedtemplates                         []                 []              [create delete
deletecollection get list patch update watch get list watch]
  routes                                     []                 []              [create delete deletecollection
get list patch update watch get list watch]
  templateconfigs                            []                 []              [create delete
deletecollection get list patch update watch get list watch]
  templateinstances                          []                 []              [create delete
deletecollection get list patch update watch get list watch]
  templates                                  []                 []              [create delete
deletecollection get list patch update watch get list watch]
  deploymentconfigs.apps.openshift.io/scale  []                 []              [create delete
deletecollection get list patch update watch get list watch]
  deploymentconfigs.apps.openshift.io        []                 []              [create delete
deletecollection get list patch update watch get list watch]
  buildconfigs.build.openshift.io/webhooks   []                 []              [create delete
deletecollection get list patch update watch get list watch]
  buildconfigs.build.openshift.io            []                 []              [create delete
deletecollection get list patch update watch get list watch]
  buildlogs.build.openshift.io               []                 []              [create delete
deletecollection get list patch update watch get list watch]
  imagestreamimages.image.openshift.io       []                 []              [create delete
deletecollection get list patch update watch get list watch]
  imagestreammappings.image.openshift.io     []                 []              [create delete
deletecollection get list patch update watch get list watch]
```

```
  imagestreamtags.image.openshift.io                []             []          [create delete
deletecollection get list patch update watch get list watch]
  routes.route.openshift.io                         []             []          [create delete
deletecollection get list patch update watch get list watch]
  processedtemplates.template.openshift.io          []             []          [create delete
deletecollection get list patch update watch get list watch]
  templateconfigs.template.openshift.io             []             []          [create delete
deletecollection get list patch update watch get list watch]
  templateinstances.template.openshift.io           []             []          [create delete
deletecollection get list patch update watch get list watch]
  templates.template.openshift.io                   []             []          [create delete
deletecollection get list patch update watch get list watch]
  serviceaccounts                                   []             []          [create delete
deletecollection get list patch update watch impersonate create delete deletecollection patch
update get list watch]
  imagestreams/secrets                              []             []          [create delete
deletecollection get list patch update watch]
  rolebindings                                      []             []          [create delete
deletecollection get list patch update watch]
  roles                                             []             []          [create delete deletecollection
get list patch update watch]
  rolebindings.authorization.openshift.io           []             []          [create delete
deletecollection get list patch update watch]
  roles.authorization.openshift.io                  []             []          [create delete
deletecollection get list patch update watch]
  imagestreams.image.openshift.io/secrets           []             []          [create delete
deletecollection get list patch update watch]
  rolebindings.rbac.authorization.k8s.io            []             []          [create delete
deletecollection get list patch update watch]
  roles.rbac.authorization.k8s.io                   []             []          [create delete
deletecollection get list patch update watch]
  networkpolicies.extensions                        []             []          [create delete
deletecollection patch update create delete deletecollection get list patch update watch get
list watch]
  networkpolicies.networking.k8s.io                 []             []          [create delete
deletecollection patch update create delete deletecollection get list patch update watch get
list watch]
  configmaps                                        []             []          [create delete
deletecollection patch update get list watch]
  endpoints                                         []             []          [create delete
deletecollection patch update get list watch]
  persistentvolumeclaims                            []             []          [create delete
deletecollection patch update get list watch]
  pods                                              []             []          [create delete deletecollection
patch update get list watch]
  replicationcontrollers/scale                      []             []          [create delete
deletecollection patch update get list watch]
  replicationcontrollers                            []             []          [create delete
deletecollection patch update get list watch]
  services                                          []             []          [create delete deletecollection
patch update get list watch]
  daemonsets.apps                                   []             []          [create delete
deletecollection patch update get list watch]
  deployments.apps/scale                            []             []          [create delete
deletecollection patch update get list watch]
  deployments.apps                                  []             []          [create delete
```

```
deletecollection patch update get list watch]
  replicasets.apps/scale                                [] [] [create delete
deletecollection patch update get list watch]
  replicasets.apps                                      [] [] [create delete
deletecollection patch update get list watch]
  statefulsets.apps/scale                               [] [] [create delete
deletecollection patch update get list watch]
  statefulsets.apps                                     [] [] [create delete
deletecollection patch update get list watch]
  horizontalpodautoscalers.autoscaling                  [] [] [create delete
deletecollection patch update get list watch]
  cronjobs.batch                                        [] [] [create delete
deletecollection patch update get list watch]
  jobs.batch                                            [] [] [create delete
deletecollection patch update get list watch]
  daemonsets.extensions                                 [] [] [create delete
deletecollection patch update get list watch]
  deployments.extensions/scale                          [] [] [create delete
deletecollection patch update get list watch]
  deployments.extensions                                [] [] [create delete
deletecollection patch update get list watch]
  ingresses.extensions                                  [] [] [create delete
deletecollection patch update get list watch]
  replicasets.extensions/scale                          [] [] [create delete
deletecollection patch update get list watch]
  replicasets.extensions                                [] [] [create delete
deletecollection patch update get list watch]
  replicationcontrollers.extensions/scale               [] [] [create delete
deletecollection patch update get list watch]
  poddisruptionbudgets.policy                           [] [] [create delete
deletecollection patch update get list watch]
  deployments.apps/rollback                             [] [] [create delete
deletecollection patch update]
  deployments.extensions/rollback                       [] [] [create delete
deletecollection patch update]
  catalogsources.operators.coreos.com                   [] [] [create update
patch delete get list watch]
  clusterserviceversions.operators.coreos.com           [] [] [create update
patch delete get list watch]
  installplans.operators.coreos.com                     [] [] [create update
patch delete get list watch]
  packagemanifests.operators.coreos.com                 [] [] [create update
patch delete get list watch]
  subscriptions.operators.coreos.com                    [] [] [create update
patch delete get list watch]
  buildconfigs/instantiate                              [] [] [create]
  buildconfigs/instantiatebinary                        [] [] [create]
  builds/clone                                          [] [] [create]
  deploymentconfigrollbacks                             [] [] [create]
  deploymentconfigs/instantiate                         [] [] [create]
  deploymentconfigs/rollback                            [] [] [create]
  imagestreamimports                                    [] [] [create]
  localresourceaccessreviews                            [] [] [create]
  localsubjectaccessreviews                             [] [] [create]
  podsecuritypolicyreviews                              [] [] [create]
  podsecuritypolicyselfsubjectreviews                   [] [] [create]
```

```
  podsecuritypolicysubjectreviews                    []              []              [create]
  resourceaccessreviews                              []              []              [create]
  routes/custom-host                                 []              []              [create]
  subjectaccessreviews                               []              []              [create]
  subjectrulesreviews                                []              []              [create]
  deploymentconfigrollbacks.apps.openshift.io        []              []              [create]
  deploymentconfigs.apps.openshift.io/instantiate    []              []              [create]
  deploymentconfigs.apps.openshift.io/rollback       []              []              [create]
  localsubjectaccessreviews.authorization.k8s.io     []              []              [create]
  localresourceaccessreviews.authorization.openshift.io  []          []              [create]
  localsubjectaccessreviews.authorization.openshift.io   []          []              [create]
  resourceaccessreviews.authorization.openshift.io   []              []              [create]
  subjectaccessreviews.authorization.openshift.io    []              []              [create]
  subjectrulesreviews.authorization.openshift.io     []              []              [create]
  buildconfigs.build.openshift.io/instantiate        []              []              [create]
  buildconfigs.build.openshift.io/instantiatebinary  []              []              [create]
  builds.build.openshift.io/clone                    []              []              [create]
  imagestreamimports.image.openshift.io              []              []              [create]
  routes.route.openshift.io/custom-host              []              []              [create]
  podsecuritypolicyreviews.security.openshift.io     []              []              [create]
  podsecuritypolicyselfsubjectreviews.security.openshift.io  []      []              [create]
  podsecuritypolicysubjectreviews.security.openshift.io      []      []              [create]
  jenkins.build.openshift.io                         []              []              [edit view view admin
edit view]
  builds                                             []              []              [get create delete
deletecollection get list patch update watch get list watch]
  builds.build.openshift.io                          []              []              [get create delete
deletecollection get list patch update watch get list watch]
  projects                                           []              []              [get delete get delete get patch
update]
  projects.project.openshift.io                      []              []              [get delete get delete
get patch update]
  namespaces                                         []              []              [get get list watch]
  pods/attach                                        []              []              [get list watch create delete
deletecollection patch update]
  pods/exec                                          []              []              [get list watch create delete
deletecollection patch update]
  pods/portforward                                   []              []              [get list watch create
delete deletecollection patch update]
  pods/proxy                                         []              []              [get list watch create delete
deletecollection patch update]
  services/proxy                                     []              []              [get list watch create delete
deletecollection patch update]
  routes/status                                      []              []              [get list watch update]
  routes.route.openshift.io/status                   []              []              [get list watch update]
  appliedclusterresourcequotas                       []              []              [get list watch]
  bindings                                           []              []              [get list watch]
  builds/log                                         []              []              [get list watch]
  deploymentconfigs/log                              []              []              [get list watch]
  deploymentconfigs/status                           []              []              [get list watch]
  events                                             []              []              [get list watch]
  imagestreams/status                                []              []              [get list watch]
  limitranges                                        []              []              [get list watch]
  namespaces/status                                  []              []              [get list watch]
  pods/log                                           []              []              [get list watch]
  pods/status                                        []              []              [get list watch]
```

```
  replicationcontrollers/status                          []              []              [get list watch]
  resourcequotas/status                                  []              []              [get list watch]
  resourcequotas                                         []              []              [get list watch]
  resourcequotausages                                    []              []              [get list watch]
  rolebindingrestrictions                                []              []              [get list watch]
  deploymentconfigs.apps.openshift.io/log                []              []              [get list watch]
  deploymentconfigs.apps.openshift.io/status             []              []              [get list watch]
  controllerrevisions.apps                               []              []              [get list watch]
  rolebindingrestrictions.authorization.openshift.io     []              []              [get list watch]
  builds.build.openshift.io/log                          []              []              [get list watch]
  imagestreams.image.openshift.io/status                 []              []              [get list watch]
  appliedclusterresourcequotas.quota.openshift.io        []              []              [get list watch]
  imagestreams/layers                                    []              []              [get update get]
  imagestreams.image.openshift.io/layers                 []              []              [get update get]
  builds/details                                         []              []              [update]
  builds.build.openshift.io/details                      []              []              [update]


Name:         basic-user
Labels:       <none>
Annotations:  openshift.io/description: A user that can get basic information about projects.
              rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
 Resources                                       Non-Resource URLs  Resource Names  Verbs
  ---------                                      -----------------  --------------  -----
  selfsubjectrulesreviews                        []                 []              [create]
  selfsubjectaccessreviews.authorization.k8s.io  []                 []              [create]
  selfsubjectrulesreviews.authorization.openshift.io  []           []              [create]
  clusterroles.rbac.authorization.k8s.io         []                 []              [get list watch]
  clusterroles                                   []                 []              [get list]
  clusterroles.authorization.openshift.io        []                 []              [get list]
  storageclasses.storage.k8s.io                  []                 []              [get list]
  users                                          []                 [~]             [get]
  users.user.openshift.io                        []                 [~]             [get]
  projects                                       []                 []              [list watch]
  projects.project.openshift.io                  []                 []              [list watch]
  projectrequests                                []                 []              [list]
  projectrequests.project.openshift.io           []                 []              [list]

Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
Resources  Non-Resource URLs  Resource Names  Verbs
---------  -----------------  --------------  -----
*.*        []                 []              [*]
           [*]                []              [*]

...
```

2. 다양한 역할에 바인딩된 사용자 및 그룹을 표시하는 현재 클러스터 역할 바인딩 집합을 보려면 다음을 수행하십시오.

```
$ oc describe clusterrolebinding.rbac
```

출력 예

```
Name:         alertmanager-main
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  alertmanager-main
Subjects:
  Kind           Name              Namespace
  ----           ----              ---------
  ServiceAccount  alertmanager-main  openshift-monitoring


Name:         basic-users
Labels:       <none>
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  basic-user
Subjects:
  Kind   Name                Namespace
  ----   ----                ---------
  Group  system:authenticated


Name:         cloud-credential-operator-rolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  cloud-credential-operator-role
Subjects:
  Kind           Name     Namespace
  ----           ----     ---------
  ServiceAccount  default  openshift-cloud-credential-operator


Name:         cluster-admin
Labels:       kubernetes.io/bootstrapping=rbac-defaults
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
  Kind   Name            Namespace
  ----   ----            ---------
  Group  system:masters


Name:         cluster-admins
Labels:       <none>
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  cluster-admin
Subjects:
```

```
  Kind   Name                Namespace
  ----   ----                ---------
  Group  system:cluster-admins
  User   system:admin


Name:         cluster-api-manager-rolebinding
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  cluster-api-manager-role
Subjects:
  Kind           Name     Namespace
  ----           ----     ---------
  ServiceAccount  default  openshift-machine-api

...
```

8.5. 로컬 역할 및 바인딩 보기

oc CLI에서 **oc describe** 명령을 사용하여 로컬 역할 및 바인딩을 볼 수 있습니다.

사전 요구 사항

- **oc** CLI를 설치합니다.
- 로컬 역할 및 바인딩을 볼 수 있는 권한을 얻습니다.
  - **cluster-admin** 기본 클러스터 역할이 클러스터 전체에서 바인딩된 사용자는 로컬 역할 및 바인딩 보기를 포함하여 모든 리소스에 대해 모든 작업을 수행할 수 있습니다.
  - **admin** 기본 클러스터 역할이 로컬로 바인딩된 사용자는 해당 프로젝트의 역할 및 바인딩을 보고 관리할 수 있습니다.

절차

1. 현재 프로젝트의 다양한 역할에 바인딩된 사용자 및 그룹을 표시하는 현재의 로컬 역할 바인딩 집합을 보려면 다음을 실행합니다.

```
$ oc describe rolebinding.rbac
```

2. 다른 프로젝트에 대한 로컬 역할 바인딩을 보려면 명령에 **-n** 플래그를 추가합니다.

```
$ oc describe rolebinding.rbac -n joe-project
```

출력 예

```
Name:         admin
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  admin
```

```
Subjects:
  Kind  Name       Namespace
  ----  ----       ---------
  User  kube:admin


Name:         system:deployers
Labels:       <none>
Annotations:  openshift.io/description:
              Allows deploymentconfigs in this namespace to rollout pods in
              this namespace.  It is auto-managed by a controller; remove
              subjects to disa...
Role:
  Kind:  ClusterRole
  Name:  system:deployer
Subjects:
  Kind           Name      Namespace
  ----           ----      ---------
  ServiceAccount  deployer  joe-project


Name:         system:image-builders
Labels:       <none>
Annotations:  openshift.io/description:
              Allows builds in this namespace to push images to this
              namespace.  It is auto-managed by a controller; remove subjects
              to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-builder
Subjects:
  Kind           Name     Namespace
  ----           ----     ---------
  ServiceAccount  builder  joe-project


Name:         system:image-pullers
Labels:       <none>
Annotations:  openshift.io/description:
              Allows all pods in this namespace to pull images from this
              namespace.  It is auto-managed by a controller; remove subjects
              to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-puller
Subjects:
  Kind   Name                           Namespace
  ----   ----                           ---------
  Group  system:serviceaccounts:joe-project
```

8.6. 사용자 역할 추가

oc adm 관리자 CLI를 사용하여 역할 및 바인딩을 관리할 수 있습니다.

사용자 또는 그룹에 역할을 바인딩하거나 추가하면 역할에 따라 사용자 또는 그룹에 부여되는 액세스 권한이 부여됩니다. **oc adm policy** 명령을 사용하여 사용자 및 그룹에 역할을 추가하거나 사용자 및 그룹으로부터 역할을 제거할 수 있습니다.

기본 클러스터 역할을 프로젝트의 로컬 사용자 또는 그룹에 바인딩할 수 있습니다.

절차

1. 특정 프로젝트의 사용자에게 역할을 추가합니다.

```
$ oc adm policy add-role-to-user <role> <user> -n <project>
```

예를 들면 다음을 실행하여 **joe** 프로젝트의 **alice** 사용자에게 **admin** 역할을 추가할 수 있습니다.

```
$ oc adm policy add-role-to-user admin alice -n joe
```

작은 정보

다음 YAML을 적용하여 사용자에게 역할을 추가할 수도 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: admin-0
  namespace: joe
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: admin
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: alice
```

2. 로컬 역할 바인딩을 보고 출력에 추가되었는지 확인합니다.

```
$ oc describe rolebinding.rbac -n <project>
```

예를 들어, **joe** 프로젝트의 로컬 역할 바인딩을 보려면 다음을 수행합니다.

```
$ oc describe rolebinding.rbac -n joe
```

출력 예

```
Name:         admin
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  admin
Subjects:
  Kind  Name        Namespace
  ----  ----        ---------
```

```
  User  kube:admin


Name:         admin-0
Labels:       <none>
Annotations:  <none>
Role:
  Kind:  ClusterRole
  Name:  admin
Subjects:
  Kind  Name   Namespace
  ----  ----   ---------
  User  alice  (1)


Name:         system:deployers
Labels:       <none>
Annotations:  openshift.io/description:
              Allows deploymentconfigs in this namespace to rollout pods in
              this namespace.  It is auto-managed by a controller; remove
              subjects to disa...
Role:
  Kind:  ClusterRole
  Name:  system:deployer
Subjects:
  Kind           Name      Namespace
  ----           ----      ---------
  ServiceAccount  deployer  joe


Name:         system:image-builders
Labels:       <none>
Annotations:  openshift.io/description:
              Allows builds in this namespace to push images to this
              namespace.  It is auto-managed by a controller; remove subjects
              to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-builder
Subjects:
  Kind           Name     Namespace
  ----           ----     ---------
  ServiceAccount  builder  joe


Name:         system:image-pullers
Labels:       <none>
Annotations:  openshift.io/description:
              Allows all pods in this namespace to pull images from this
              namespace.  It is auto-managed by a controller; remove subjects
              to disable.
Role:
  Kind:  ClusterRole
  Name:  system:image-puller
Subjects:
```

```
Kind   Name                         Namespace
----   ----                         ---------
Group  system:serviceaccounts:joe
```

1 **alice** 사용자가 **admins RoleBinding**에 추가되었습니다.

8.7. 로컬 역할 생성

프로젝트의 로컬 역할을 생성하여 이 역할을 사용자에게 바인딩할 수 있습니다.

절차

1. 프로젝트의 로컬 역할을 생성하려면 다음 명령을 실행합니다.

```
$ oc create role <name> --verb=<verb> --resource=<resource> -n <project>
```

이 명령에서는 다음을 지정합니다.

- **<name>**: 로컬 역할 이름
- **<verb>**: 역할에 적용할 동사를 쉼표로 구분한 목록
- **<resource>**: 역할이 적용되는 리소스
- **<project>**: 프로젝트 이름

예를 들어, 사용자가 **blue** 프로젝트의 Pod를 볼 수 있는 로컬 역할을 생성하려면 다음 명령을 실행합니다.

```
$ oc create role podview --verb=get --resource=pod -n blue
```

2. 새 역할을 사용자에게 바인딩하려면 다음 명령을 실행합니다.

```
$ oc adm policy add-role-to-user podview user2 --role-namespace=blue -n blue
```

8.8. 클러스터 역할 생성

클러스터 역할을 만들 수 있습니다.

절차

1. 클러스터 역할을 만들려면 다음 명령을 실행합니다.

```
$ oc create clusterrole <name> --verb=<verb> --resource=<resource>
```

이 명령에서는 다음을 지정합니다.

- **<name>**: 로컬 역할 이름
- **<verb>**: 역할에 적용할 동사를 쉼표로 구분한 목록
- **<resource>**: 역할이 적용되는 리소스

예를 들어 사용자가 pod를 볼 수 있는 클러스터 역할을 만들려면 다음 명령을 실행합니다.

```
$ oc create clusterrole podviewonly --verb=get --resource=pod
```

8.9. 로컬 역할 바인딩 명령

다음 작업을 사용하여 로컬 역할 바인딩에 대한 사용자 또는 그룹의 연결된 역할을 관리하는 경우, **-n** 플래그를 사용하여 프로젝트를 지정할 수 있습니다. 지정하지 않으면 현재 프로젝트가 사용됩니다.

다음 명령을 사용하여 로컬 RBAC를 관리할 수 있습니다.

표 8.1. 로컬 역할 바인딩 작업

명령	설명
$ oc adm policy who-can *<verb> <resource>*	리소스에 작업을 수행할 수 있는 사용자를 나타냅니다.
$ oc adm policy add-role-to-user *<role> <username>*	현재 프로젝트에서 지정된 사용자에게 지정된 역할을 바인딩합니다.
$ oc adm policy remove-role-from-user *<role> <username>*	현재 프로젝트에서 지정된 사용자로부터 지정된 역할을 제거합니다.
$ oc adm policy remove-user *<username>*	현재 프로젝트에서 지정된 사용자 및 해당 사용자의 역할을 모두 제거합니다.
$ oc adm policy add-role-to-group *<role> <groupname>*	현재 프로젝트에서 지정된 그룹에 지정된 역할을 바인딩합니다.
$ oc adm policy remove-role-from-group *<role> <groupname>*	현재 프로젝트에서 지정된 그룹의 지정된 역할을 제거합니다.
$ oc adm policy remove-group *<groupname>*	현재 프로젝트에서 지정된 그룹과 해당 그룹의 역할을 모두 제거합니다.

8.10. 클러스터 역할 바인딩 명령

다음 작업을 사용하여 클러스터 역할 바인딩을 관리할 수도 있습니다. 클러스터 역할 바인딩에 네임스페이스가 아닌 리소스가 사용되므로 **-n** 플래그가 해당 작업에 사용되지 않습니다.

표 8.2. 클러스터 역할 바인딩 작업

명령	설명
$ oc adm policy add-cluster-role-to-user *<role> <username>*	클러스터의 모든 프로젝트에 대해 지정된 사용자에게 지정된 역할을 바인딩합니다.
$ oc adm policy remove-cluster-role-from-user *<role> <username>*	클러스터의 모든 프로젝트에 대해 지정된 사용자로부터 지정된 역할을 제거합니다.

명령	설명
$ oc adm policy add-cluster-role-to-group *<role> <groupname>*	클러스터의 모든 프로젝트에 대해 지정된 역할을 지정된 그룹에 바인딩합니다.
$ oc adm policy remove-cluster-role-from-group *<role> <groupname>*	클러스터의 모든 프로젝트에 대해 지정된 그룹에서 지정된 역할을 제거합니다.

8.11. 클러스터 관리자 생성

클러스터 리소스 수정과 같은 OpenShift Container Platform 클러스터에서 관리자 수준 작업을 수행하려면 **cluster-admin** 역할이 필요합니다.

사전 요구 사항

- 클러스터 관리자로 정의할 사용자를 생성해야 합니다.

절차

- 사용자를 클러스터 관리자로 정의합니다.

```
$ oc adm policy add-cluster-role-to-user cluster-admin <user>
```

9장. KUBEADMIN 사용자 제거

9.1. KUBEADMIN 사용자

OpenShift Container Platform에서는 설치 프로세스가 완료되면 클러스터 관리자 **kubeadmin**을 생성합니다.

이 사용자는 **cluster-admin** 역할이 자동으로 적용되며 클러스터의 루트 사용자로 취급됩니다. 암호는 동적으로 생성되며 OpenShift Container Platform 환경에서 고유합니다. 설치가 완료되면 설치 프로그램의 출력에 암호가 제공됩니다. 예를 들면 다음과 같습니다.

```
INFO Install complete!
INFO Run 'export KUBECONFIG=<your working directory>/auth/kubeconfig' to manage the cluster
with 'oc', the OpenShift CLI.
INFO The cluster is ready when 'oc login -u kubeadmin -p <provided>' succeeds (wait a few minutes).
INFO Access the OpenShift web-console here: https://console-openshift-
console.apps.demo1.openshift4-beta-abcorp.com
INFO Login to the console with user: kubeadmin, password: <provided>
```

9.2. KUBEADMIN 사용자 제거

ID 공급자를 정의하고 새 **cluster-admin** 사용자를 만든 다음 **kubeadmin**을 제거하여 클러스터 보안을 강화할 수 있습니다.

주의

다른 사용자가 **cluster-admin**이 되기 전에 이 절차를 수행하는 경우 OpenShift Container Platform을 다시 설치해야 합니다. 이 명령은 취소할 수 없습니다.

사전 요구 사항

- 하나 이상의 ID 공급자를 구성해야 합니다.
- 사용자에게 **cluster-admin** 역할을 추가해야 합니다.
- 관리자로 로그인해야 합니다.

절차

- **kubeadmin** 시크릿을 제거합니다.

```
$ oc delete secrets kubeadmin -n kube-system
```

10장. 서비스 계정 이해 및 생성

10.1. 서비스 계정 개요

서비스 계정은 구성 요소가 API에 직접 액세스할 수 있는 OpenShift Container Platform 계정입니다. 서비스 계정은 각 프로젝트 내에 존재하는 API 오브젝트입니다. 서비스 계정을 사용하면 일반 사용자의 자격 증명을 공유하지 않고도 API 액세스 권한을 유연하게 제어할 수 있습니다.

OpenShift Container Platform CLI 또는 웹 콘솔을 사용하면 API 토큰을 통해 API에 대한 인증이 수행됩니다. 일반 사용자의 자격 증명을 사용하지 않고도 API에 액세스할 수 있도록 구성 요소를 서비스 계정과 연결할 수 있습니다. 예를 들어, 서비스 계정을 사용하면 다음을 수행할 수 있습니다.

- Pod를 생성하거나 삭제하기 위해 복제 컨트롤러를 통해 API를 호출합니다.
- 검색을 위해 컨테이너 내부 애플리케이션을 통해 API를 호출합니다.
- 모니터링 또는 통합을 위해 외부 애플리케이션을 통해 API를 호출합니다.

각 서비스 계정의 사용자 이름은 프로젝트와 이름에서 파생됩니다.

```
system:serviceaccount:<project>:<name>
```

모든 서비스 계정은 다음 두 그룹의 멤버이기도 합니다.

그룹	설명
system:serviceaccounts	시스템의 모든 서비스 계정이 포함됩니다.
system:serviceaccounts:<project>	지정된 프로젝트의 모든 서비스 계정이 포함됩니다.

각 서비스 계정에는 다음 두 가지 시크릿이 자동으로 포함됩니다.

- API 토큰
- OpenShift Container Registry 자격 증명

생성된 API 토큰 및 레지스트리 인증 정보는 만료되지 않지만 시크릿을 삭제하여 무효화할 수 있습니다. 시크릿을 삭제하면 새로운 시크릿이 자동으로 생성되어 대체됩니다.

10.2. 서비스 계정 생성

프로젝트에서 서비스 계정을 생성하고 역할에 바인딩하여 권한을 부여할 수 있습니다.

절차

1. 선택 사항: 현재 프로젝트에서 서비스 계정을 보려면 다음을 수행합니다.

```
$ oc get sa
```

출력 예

```
NAME      SECRETS   AGE
builder   2         2d
default   2         2d
deployer  2         2d
```

2. 현재 프로젝트에서 새 서비스 계정을 생성하려면 다음을 수행합니다.

```
$ oc create sa <service_account_name> 1
```

1 다른 프로젝트에서 서비스 계정을 생성하려면 **-n <project_name>**을 지정합니다.

출력 예

```
serviceaccount "robot" created
```

작은 정보

다음 YAML을 적용하여 서비스 계정을 생성할 수도 있습니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <service_account_name>
  namespace: <current_project>
```

3. 선택 사항: 서비스 계정의 시크릿을 확인합니다.

```
$ oc describe sa robot
```

출력 예

```
Name:  robot
Namespace: project1
Labels:  <none>
Annotations: <none>

Image pull secrets: robot-dockercfg-qzbhb

Mountable secrets:  robot-token-f4khf
                    robot-dockercfg-qzbhb

Tokens:             robot-token-f4khf
                    robot-token-z8h44
```

10.3. 서비스 계정에 역할을 부여하는 예

일반 사용자 계정에 역할을 부여하는 것과 같은 방식으로 서비스 계정에 역할을 부여할 수 있습니다.

- 현재 프로젝트의 서비스 계정을 수정할 수 있습니다. 예를 들어, **top-secret** 개체에서 **robot** 서비스 계정에 **view** 역할을 추가하려면 다음을 수행합니다.

```
$ oc policy add-role-to-user view system:serviceaccount:top-secret:robot
```

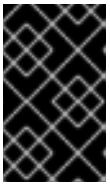
작은 정보

다음 YAML을 적용하여 역할을 추가할 수도 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: view
  namespace: top-secret
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- kind: ServiceAccount
  name: robot
  namespace: top-secret
```

- 프로젝트의 특정 서비스 계정에 대한 액세스 권한을 부여할 수도 있습니다. 예를 들어 서비스 계정이 속하는 프로젝트에서 **-z** 플래그를 사용하고 **<service_account_name>**을 지정합니다.

```
$ oc policy add-role-to-user <role_name> -z <service_account_name>
```



중요

프로젝트의 특정 서비스 계정에 대한 액세스 권한을 부여하려면 **-z** 플래그를 사용합니다. 이 플래그를 사용하면 오타를 방지하고 지정된 서비스 계정에만 액세스 권한을 부여할 수 있습니다.

작은 정보

다음 YAML을 적용하여 역할을 추가할 수도 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: <rolebinding_name>
  namespace: <current_project_name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: <role_name>
subjects:
- kind: ServiceAccount
  name: <service_account_name>
  namespace: <current_project_name>
```

- 다른 네임스페이스를 수정하려면 다음 예제와 같이 **-n** 옵션을 사용하여 적용되는 프로젝트 네임스페이스를 나타낼 수 있습니다.

- 예를 들어 모든 프로젝트의 모든 서비스 계정에서 **my-project** 프로젝트의 리소스를 볼 수 있도록 하려면 다음을 실행합니다.

```
$ oc policy add-role-to-group view system:serviceaccounts -n my-project
```

작은 정보

다음 YAML을 적용하여 역할을 추가할 수도 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: view
  namespace: my-project
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: view
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts
```

- **managers** 프로젝트의 모든 서비스 계정에서 **my-project** 프로젝트의 리소스를 편집할 수 있도록 하려면 다음을 실행합니다.

```
$ oc policy add-role-to-group edit system:serviceaccounts:managers -n my-project
```

작은 정보

다음 YAML을 적용하여 역할을 추가할 수도 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: edit
  namespace: my-project
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: edit
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:serviceaccounts:managers
```

11장. 애플리케이션에서 서비스 계정 사용

11.1. 서비스 계정 개요

서비스 계정은 구성 요소가 API에 직접 액세스할 수 있는 OpenShift Container Platform 계정입니다. 서비스 계정은 각 프로젝트 내에 존재하는 API 오브젝트입니다. 서비스 계정을 사용하면 일반 사용자의 자격 증명을 공유하지 않고도 API 액세스 권한을 유연하게 제어할 수 있습니다.

OpenShift Container Platform CLI 또는 웹 콘솔을 사용하면 API 토큰을 통해 API에 대한 인증이 수행됩니다. 일반 사용자의 자격 증명을 사용하지 않고도 API에 액세스할 수 있도록 구성 요소를 서비스 계정과 연결할 수 있습니다. 예를 들어, 서비스 계정을 사용하면 다음을 수행할 수 있습니다.

- Pod를 생성하거나 삭제하기 위해 복제 컨트롤러를 통해 API를 호출합니다.
- 검색을 위해 컨테이너 내부 애플리케이션을 통해 API를 호출합니다.
- 모니터링 또는 통합을 위해 외부 애플리케이션을 통해 API를 호출합니다.

각 서비스 계정의 사용자 이름은 프로젝트와 이름에서 파생됩니다.

```
system:serviceaccount:<project>:<name>
```

모든 서비스 계정은 다음 두 그룹의 멤버이기도 합니다.

그룹	설명
system:serviceaccounts	시스템의 모든 서비스 계정이 포함됩니다.
system:serviceaccounts:<project>	지정된 프로젝트의 모든 서비스 계정이 포함됩니다.

각 서비스 계정에는 다음 두 가지 시크릿이 자동으로 포함됩니다.

- API 토큰
- OpenShift Container Registry 자격 증명

생성된 API 토큰 및 레지스트리 인증 정보는 만료되지 않지만 시크릿을 삭제하여 무효화할 수 있습니다. 시크릿을 삭제하면 새로운 시크릿이 자동으로 생성되어 대체됩니다.

11.2. 기본 서비스 계정

OpenShift Container Platform 클러스터에는 클러스터 관리를 위해 기본 서비스 계정이 포함되어 있으며, 프로젝트마다 더 많은 서비스 계정이 생성됩니다.

11.2.1. 기본 클러스터 서비스 계정

다양한 인프라 컨트롤러가 서비스 계정 자격 증명을 사용하여 실행됩니다. 다음 서비스 계정은 서버 시작 시 OpenShift Container Platform 인프라 프로젝트(**openshift-infra**)에서 생성되며, 클러스터 전체에서 다음 역할이 제공됩니다.

서비스 계정	설명
replication-controller	**system:replication-controller** 역할이 할당됨
deployment-controller	**system:deployment-controller** 역할이 할당됨
build-controller	**system:build-controller** 역할이 할당되었습니다. 또한 권한 있는 빌드 pod를 생성하기 위해 권한 있는 보안 컨텍스트 제약 조건에 **build-controller** 서비스 계정이 포함되어 있습니다.

11.2.2. 기본 프로젝트 서비스 계정 및 역할

프로젝트당 3개의 서비스 계정이 자동으로 생성됩니다.

서비스 계정	사용법
builder	빌드 Pod에서 사용합니다. 내부 Docker 레지스트리를 사용하여 프로젝트의 이미지 스트림으로 이미지를 밀어 넣을 수 있는 **system:image-builder** 역할이 제공됩니다.
deployer	배포 Pod에서 사용하며 **system:deployer** 역할을 지정하는 경우 프로젝트에서 복제 컨트롤러 및 Pod를 보고 수정할 수 있습니다.
default	다른 서비스 계정을 지정하지 않는 한 기타 모든 Pod를 실행하는 데 사용합니다.

프로젝트의 모든 서비스 계정에는 **system:image-puller** 역할이 부여되어 내부 컨테이너 이미지 레지스트리를 사용하여 프로젝트의 모든 이미지 스트림에서 이미지를 가져올 수 있습니다.

11.3. 서비스 계정 생성

프로젝트에서 서비스 계정을 생성하고 역할에 바인딩하여 권한을 부여할 수 있습니다.

절차

1. 선택 사항: 현재 프로젝트에서 서비스 계정을 보려면 다음을 수행합니다.

```
$ oc get sa
```

출력 예

```
NAME       SECRETS   AGE
builder    2         2d
default    2         2d
deployer   2         2d
```

2. 현재 프로젝트에서 새 서비스 계정을 생성하려면 다음을 수행합니다.

```
$ oc create sa <service_account_name> 1
```

1 다른 프로젝트에서 서비스 계정을 생성하려면 **-n <project_name>**을 지정합니다.

출력 예

```
serviceaccount "robot" created
```

작은 정보

다음 YAML을 적용하여 서비스 계정을 생성할 수도 있습니다.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: <service_account_name>
  namespace: <current_project>
```

3. 선택 사항: 서비스 계정의 시크릿을 확인합니다.

```
$ oc describe sa robot
```

출력 예

```
Name:  robot
Namespace: project1
Labels:  <none>
Annotations: <none>

Image pull secrets: robot-dockercfg-qzbhb

Mountable secrets:  robot-token-f4khf
                    robot-dockercfg-qzbhb

Tokens:             robot-token-f4khf
                    robot-token-z8h44
```

11.4. 서비스 계정의 인증 정보를 외부에서 사용

API 인증이 필요한 외부 애플리케이션에 서비스 계정의 토큰을 배포할 수 있습니다.

이미지를 가져오려면 인증된 사용자에게 요청된 **imagestreams/layers**에 대한 **get** 권한이 있어야 합니다. 이미지를 밀어 넣으려면 인증된 사용자에게 요청된 **imagestreams/layers**에 대한 **update** 권한이 있어야 합니다.

기본적으로 프로젝트의 모든 서비스 계정에는 해당 프로젝트의 이미지를 가져올 수 있는 권한이 있고, **builder** 서비스 계정에는 해당 프로젝트에 이미지를 밀어 넣을 권한이 있습니다.

절차

1. 서비스 계정의 API 토큰을 확인합니다.

```
$ oc describe secret <secret_name>
```

예를 들면 다음과 같습니다.

```
$ oc describe secret robot-token-uzkbh -n top-secret
```

출력 예

```
Name:  robot-token-uzkbh
Labels:  <none>
Annotations: kubernetes.io/service-account.name=robot,kubernetes.io/service-
account.uid=49f19e2e-16c6-11e5-afdc-3c970e4b7ffe

Type: kubernetes.io/service-account-token

Data

token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

2. 가져온 토큰을 사용하여 로그인합니다.

```
$ oc login --token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

출력 예

```
Logged into "https://server:8443" as "system:serviceaccount:top-secret:robot" using the
token provided.

You don't have any projects. You can try to create a new project, by running

    $ oc new-project <projectname>
```

3. 서비스 계정으로 로그인했는지 확인합니다.

```
$ oc whoami
```

출력 예

```
system:serviceaccount:top-secret:robot
```

12장. 서비스 계정을 OAUTH 클라이언트로 사용

12.1. OAUTH 클라이언트로서의 서비스 계정

서비스 계정을 제한된 형태의 OAuth 클라이언트로 사용할 수 있습니다. 서비스 계정에서는 서비스 계정의 자체 네임스페이스 내에 있는 일부 기본 사용자 정보 및 역할 기반 권한에 액세스할 수 있는 부분적인 범위만 요청할 수 있습니다.

- **user:info**
- **user:check-access**
- **role:<any_role>:<service_account_namespace>**
- **role:<any_role>:<service_account_namespace>:!**

서비스 계정을 OAuth 클라이언트로 사용하는 경우에는 다음과 같습니다.

- **client_id**는 **system:serviceaccount:<service_account_namespace>:<service_account_name>**입니다.
- **client_secret**은 해당 서비스 계정의 API 토큰 중 하나일 수 있습니다. 예를 들면 다음과 같습니다.

```
$ oc sa get-token <service_account_name>
```

- **WWW-Authenticate** 챌린지를 가져오려면 서비스 계정의 **serviceaccounts.openshift.io/oauth-want-challenges** 주석을 **true**로 설정합니다.
- **redirect_uri**는 서비스 계정의 주석과 일치해야 합니다.

12.1.1. 서비스 계정의 URI를 OAuth 클라이언트로 리디렉션

주석 키에는 접두사 **serviceaccounts.openshift.io/oauth-redirecturi.** 또는 **serviceaccounts.openshift.io/oauth-redirectreference.**가 있어야 합니다. 예를 들면 다음과 같습니다.

```
serviceaccounts.openshift.io/oauth-redirecturi.<name>
```

가장 간단한 형식의 주석을 사용하여 유효한 리디렉션 URI를 직접 지정할 수 있습니다. 예를 들면 다음과 같습니다.

```
"serviceaccounts.openshift.io/oauth-redirecturi.first":  "https://example.com"
"serviceaccounts.openshift.io/oauth-redirecturi.second": "https://other.com"
```

위 예에서 **first** 및 **second**라는 접미사는 두 개의 유효한 리디렉션 URI를 구분하는 데 사용됩니다.

복잡한 구성에서는 정적 리디렉션 URI로는 충분하지 않을 수 있습니다. 예를 들면 특정 경로의 모든 Ingress를 유효한 것으로 간주해야 할 수 있습니다. 이 경우 **serviceaccounts.openshift.io/oauth-redirectreference** 접두사를 통한 동적 리디렉션 URI가 작동합니다.

예를 들면 다음과 같습니다.

```
"serviceaccounts.openshift.io/oauth-redirectreference.first": "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"jenkins\"}}"
```

-

이 주석의 값은 직렬화된 JSON 데이터를 포함하므로 확장된 형식으로 쉽게 볼 수 있습니다.

```
{
  "kind": "OAuthRedirectReference",
  "apiVersion": "v1",
  "reference": {
    "kind": "Route",
    "name": "jenkins"
  }
}
```

이제 **OAuthRedirectReference**를 통해 **jenkins**라는 경로를 참조할 수 있음을 확인할 수 있습니다. 따라서 해당 경로의 모든 Ingress가 이제 유효한 것으로 간주됩니다. **OAuthRedirectReference**의 전체 사양은 다음과 같습니다.

```
{
  "kind": "OAuthRedirectReference",
  "apiVersion": "v1",
  "reference": {
    "kind": ..., (1)
    "name": ..., (2)
    "group": ... (3)
  }
}
```

1. **kind**는 참조되는 오브젝트의 유형을 나타냅니다. 현재는 **route**만 지원됩니다.
2. **name**은 오브젝트의 이름을 나타냅니다. 오브젝트는 서비스 계정과 동일한 네임스페이스에 있어야 합니다.
3. **group**은 오브젝트 그룹을 나타냅니다. 경로 그룹이 빈 문자열이므로 이 필드를 비워둡니다.

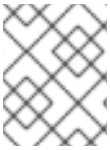
두 주석 접두사를 결합하여 참조 오브젝트에서 제공하는 데이터를 덮어쓸 수 있습니다. 예를 들면 다음과 같습니다.

```
"serviceaccounts.openshift.io/oauth-redirecturi.first":  "custompath"
"serviceaccounts.openshift.io/oauth-redirectreference.first": "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"jenkins\"}}"
```

first 접미사는 주석을 연결하는 데 사용됩니다. **jenkins** 경로에 **https://example.com**의 Ingress가 있다고 가정하면 이제 **https://example.com/custompath**는 유효한 것으로 간주되지만 **https://example.com**은 유효한 것으로 간주되지 않습니다. 데이터 덮어쓰기를 부분적으로 제공하는 형식은 다음과 같습니다.

유형	구문
스키마	"https://"
호스트 이름	"//website.com"
포트	"//:8000"

유형	구문
경로	"examplepath"



참고

호스트 이름 덮어쓰기를 지정하면 참조된 오브젝트의 호스트 이름 데이터가 교체되므로 바람직한 동작이 아닙니다.

위 구문의 모든 조합은 다음과 같은 형식을 사용하여 결합할 수 있습니다.

<scheme:>//<hostname><:port>/<path>

유연성을 개선하기 위해 동일한 오브젝트를 두 번 이상 참조할 수 있습니다.

```
"serviceaccounts.openshift.io/oauth-redirecturi.first":  "custompath"
"serviceaccounts.openshift.io/oauth-redirectreference.first": "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"jenkins\"}}"
"serviceaccounts.openshift.io/oauth-redirecturi.second":  "//:8000"
"serviceaccounts.openshift.io/oauth-redirectreference.second": "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"jenkins\"}}"
```

jenkins라는 경로에 **https://example.com** Ingress가 있다고 가정하면 **https://example.com:8000** 및 **https://example.com/custompath**가 모두 유효한 것으로 간주됩니다.

정적 및 동적 주석을 동시에 사용하여 원하는 동작을 수행할 수 있습니다.

```
"serviceaccounts.openshift.io/oauth-redirectreference.first": "
{\"kind\":\"OAuthRedirectReference\",\"apiVersion\":\"v1\",\"reference\":
{\"kind\":\"Route\",\"name\":\"jenkins\"}}"
"serviceaccounts.openshift.io/oauth-redirecturi.second": "https://other.com"
```

13장. 범위 지정 토큰

13.1. 범위 지정 토큰 정보

범위가 지정된 토큰을 생성하여 일부 권한을 다른 사용자나 서비스 계정에 위임할 수 있습니다. 예를 들어, 프로젝트 관리자가 Pod 생성 권한을 위임하고자 할 수 있습니다.

범위가 지정된 토큰은 지정된 사용자로 확인되지만 범위에 따라 특정 작업으로 제한되는 토큰에 해당합니다. **cluster-admin** 역할의 사용자만 범위가 지정된 토큰을 생성할 수 있습니다.

범위는 토큰의 범위 집합을 **PolicyRules** 집합으로 변환하여 평가합니다. 그러면 요청이 해당 규칙과 대조됩니다. 추가적인 권한 확인을 위해 "일반"권한 부여자에게 전달하려면 요청 속성이 하나 이상의 범위 규칙과 일치해야 합니다.

13.1.1. 사용자 범위

사용자 범위는 지정된 사용자에 대한 정보를 얻는 데 중점을 둡니다. 의도 기반이므로 다음과 같이 규칙이 자동으로 생성됩니다.

- **user:full** - 모든 사용자 권한에 API에 대한 전체 읽기/쓰기 액세스를 허용합니다.
- **user:info** - 사용자 정보(예: 이름, 그룹)에 대한 읽기 전용 액세스를 허용합니다.
- **user:check-access** - **self-localsubjectaccessreviews** 및 **self-subjectaccessreviews**에 대한 액세스를 허용합니다. 해당 항목은 요청 오브젝트에서 빈 사용자 및 그룹을 전달하는 변수입니다.
- **user:list-projects** - 사용자가 액세스할 수 있는 프로젝트를 나열하도록 읽기 전용 액세스를 허용합니다.

13.1.2. 역할 범위

역할 범위를 사용하면 네임스페이스로 필터링되어 지정된 역할과 동일한 수준의 액세스 권한을 가질 수 있습니다.

- **role:<cluster-role name>:<namespace or * for all>** - 클러스터 역할에 따라 지정된 규칙으로 범위가 제한되지만, 지정된 네임스페이스에 한합니다.

참고

경고 사항: 이렇게 하면 액세스 권한이 에스컬레이션되지 않습니다. 시크릿, 역할 바인딩 및 역할과 같은 리소스에 액세스할 수 있는 역할이더라도 이 범위는 해당 리소스에 대한 액세스를 거부합니다. 따라서 예기치 않은 에스컬레이션을 방지할 수 있습니다. 대부분의 사람들은 **edit**와 같은 역할을 에스컬레이션되는 역할로 생각하지 않지만 시크릿에 액세스할 수 있는 경우 에스컬레이션됩니다.

- **role:<cluster-role name>:<namespace or * for all>:!** - 느낌표를 넣어 이 범위에서 액세스 권한을 에스컬레이션할 수 있다는 점을 제외하면 위의 예와 유사합니다.

14장. 바인딩된 서비스 계정 토큰 사용

바인딩된 서비스 계정 토큰을 사용하여 AWS IAM과 같은 클라우드 공급자 IAM(Identity Access Management) 서비스와의 통합을 개선할 수 있습니다.

14.1. 바인딩된 서비스 계정 토큰 정보

바인딩된 서비스 계정 토큰을 사용하여 지정된 서비스 계정 토큰에 대한 권한 범위를 제한할 수 있습니다. 이 토큰에는 오디언스와 시간이 바인딩되어 있습니다. 이를 통해 IAM 역할에 대한 서비스 계정 인증 및 Pod에 마운트된 임시 인증 정보 생성이 용이해집니다. 볼륨 프로젝션 및 TokenRequest API를 사용하여 바인딩된 서비스 계정 토큰을 요청할 수 있습니다.

14.2. 볼륨 프로젝션을 사용한 바인딩된 서비스 계정 토큰 구성

볼륨 프로젝션을 통해 바인딩된 서비스 계정 토큰을 요청하도록 pod를 구성할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있습니다.
- 서비스 계정을 생성했습니다. 이 절차에서는 서비스 계정의 이름이 **build-robot**이라고 가정합니다.

절차

1. 선택 사항: 서비스 계정 발행자를 설정합니다.
   바인딩된 토큰이 클러스터 내에서만 사용되는 경우 일반적으로 이 단계가 필요하지 않습니다.

> **주의**
>
> **serviceAccountIssuer** 필드를 업데이트하고 이미 사용 중인 바인딩 토큰이 있으면 이전 발급자 값이 있는 모든 바인딩 토큰이 무효화됩니다. 바인딩된 토큰 소유자가 발급자 변경에 대한 명시적으로 지원이 없는 경우, 소유자는 Pod를 다시 시작할 때까지 새 바인딩 토큰을 요청하지 않습니다.
>
> 클러스터의 모든 Pod를 수동으로 다시 시작하거나 롤링 노드를 다시 시작하여 모든 소유자가 새 바인딩된 토큰을 요청하도록 할 수 있습니다. 두 작업을 수행하기 전에 Kubernetes API 서버 Pod의 새 버전이 서비스 계정 발행자 변경과 함께 롤아웃될 때까지 기다립니다.

   a. **cluster Authentication** 오브젝트를 편집합니다.

```
$ oc edit authentications cluster
```

   b. **spec.serviceAccountIssuer** 필드를 원하는 서비스 계정 발행자 값으로 설정합니다.

```
spec:
  serviceAccountIssuer: https://test.default.svc 1
```

1 이 값은 바인딩된 토큰의 수신자가 토큰 서명을 확인하는 데 필요한 공개 키를 공급할 수 있는 URL이어야 합니다. 기본값은 **https://kubernetes.default.svc**입니다.

c. 파일을 저장하여 변경 사항을 적용합니다.

d. Kubernetes API 서버 포드의 새 버전이 롤아웃될 때까지 기다립니다. 모든 노드가 새 버전으로 업데이트되는 데 몇 분이 걸릴 수 있습니다. 다음 명령을 실행합니다.

```
$ oc get kubeapiserver -o=jsonpath='{range .items[0].status.conditions[?
(@.type=="NodeInstallerProgressing")]}{.reason}{"\n"}{.message}{"\n"}'
```

Kubernetes API 서버의 **NodeInstallerProgressing** 상태 조건을 검토하여 모든 노드가 최신 버전인지 확인합니다. 업데이트가 성공적으로 실행되면 출력에 **AllNodesAtLatestRevision**이 표시됩니다.

```
AllNodesAtLatestRevision
3 nodes are at revision 12 1
```

1 이 예에서 최신 버전 번호는 **12** 입니다.

출력에 다음 메시지 중 하나와 유사한 메시지가 표시되면 업데이트가 계속 진행 중입니다. 몇 분 기다린 후 다시 시도합니다.

- **3 nodes are at revision 11; 0 nodes have achieved new revision 12**
- **2 nodes are at revision 11; 1 nodes are at revision 12**

e. 선택 사항: 홀더가 롤링 노드 재시작을 수행하거나 클러스터의 모든 Pod를 수동으로 다시 시작하여 새 바인딩된 토큰을 요청하도록 강제 적용합니다.

- 롤링 노드 재시작을 수행합니다.

주의

클러스터에서 사용자 정의 워크로드를 실행 중인 경우 서비스가 중단될 수 있으므로 롤링 노드 재시작을 수행하지 않는 것이 좋습니다. 대신 클러스터의 모든 Pod를 수동으로 다시 시작합니다.

노드를 순차적으로 재시작합니다. 다음 노드를 다시 시작하기 전에 노드를 완전히 사용할 수 있을 때까지 기다립니다. *노드를 다시 예약 가능으로 드레이닝, 재시작, 표시하는 방법에 대한 지침은 정상적으로 노드 재부팅* 을 참조하십시오.

- 클러스터의 모든 Pod를 수동으로 다시 시작합니다.

주의

이 명령을 실행하면 모든 네임스페이스에서 실행 중인 모든 Pod가 삭제되므로 서비스가 중단됩니다. 이 Pod는 삭제 후 자동으로 다시 시작됩니다.

다음 명령을 실행합니다.

```
$ for I in $(oc get ns -o jsonpath='{range .items[*]} {.metadata.name}{"\n"} {end}'); \
    do oc delete pods --all -n $I; \
    sleep 1; \
    done
```

2. 볼륨 프로젝션을 통해 바인딩된 서비스 계정 토큰을 사용하도록 pod를 구성합니다.

   a. 다음 콘텐츠를 사용하여 **pod-projected-svc-token.yaml**이라는 파일을 생성합니다.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /var/run/secrets/tokens
      name: vault-token
  serviceAccountName: build-robot (1)
  volumes:
  - name: vault-token
    projected:
      sources:
      - serviceAccountToken:
          path: vault-token (2)
          expirationSeconds: 7200 (3)
          audience: vault (4)
```

(1) 기존 서비스 계정에 대한 참조입니다.

(2) 토큰을 프로젝션할 파일의 마운트 지점을 기준으로 하는 경로입니다.

(3) 필요한 경우 서비스 계정 토큰 만료 시간을 초 단위로 설정합니다. 기본값은 3600초(1시간)이며 600초(10분) 이상이어야 합니다. 토큰이 수명의 80% 이상을 경과했거나 24시간 이상 된 경우, kubelet에서 토큰을 회전합니다.

(4) 필요한 경우 의도된 토큰 오디언스를 설정합니다. 토큰 수신자는 수신자 ID가 토큰의 오디언스 클레임과 일치하는지 확인하고 일치하지 않는 경우 토큰을 거부해야 합니다. 오디언스는 기본적으로 API 서버의 식별자입니다.

b. Pod를 생성합니다.

```
$ oc create -f pod-projected-svc-token.yaml
```

kubelet은 pod를 대신하여 토큰을 요청 및 저장하고, 구성 가능한 파일 경로에 있는 pod에서 토큰을 사용할 수 있도록 설정하고, 토큰이 만료되면 토큰을 갱신합니다.

3. 바인딩된 토큰을 사용하는 애플리케이션에서는 토큰이 회전할 때 다시 로드하는 작업을 처리해야 합니다.
kubelet은 토큰이 수명의 80% 이상을 경과했거나 24시간 이상된 경우, 토큰을 회전합니다.

추가 리소스

- 노드를 정상적으로 재부팅

15장. 보안 컨텍스트 제약 조건 관리

15.1. 보안 컨텍스트 제약 조건 정보

RBAC 리소스에서 사용자 액세스 권한을 제어하는 방식과 유사하게 관리자는 SCC(*보안 컨텍스트 제약 조건*)를 사용하여 Pod에 대한 권한을 제어할 수 있습니다. 이러한 권한에는 Pod에서 수행할 수 있는 작업과 액세스할 수 있는 리소스가 포함됩니다. Pod가 시스템에 수용되려면 일련의 조건을 함께 실행해야 하는데, SCC를 사용하여 이러한 조건을 정의할 수 있습니다.

시크릿 컨텍스트 제약 조건을 통해 관리자는 다음을 제어할 수 있습니다.

- Pod에서 **allowPrivilegedContainer** 플래그를 사용하여 권한 있는 컨테이너를 실행할 수 있는지 여부입니다.
- Pod가 **allowPrivilegeEscalation** 플래그로 제한되는지 여부입니다.
- 컨테이너에서 요청할 수 있는 기능
- 호스트 디렉터리를 볼륨으로 사용
- 컨테이너의 SELinux 컨텍스트
- 컨테이너 사용자 ID
- 호스트 네임스페이스 및 네트워킹 사용
- Pod 볼륨을 보유한 **FSGroup**의 할당
- 허용되는 추가 그룹의 구성
- 컨테이너에 루트 파일 시스템에 대한 쓰기 액세스 권한이 필요한지 여부
- 볼륨 유형 사용
- 허용 가능한 **seccomp** 프로필 구성

중요

OpenShift Container Platform의 네임스페이스에 **openshift.io/run-level** 레이블을 설정하지 마십시오. 이 레이블은 내부 OpenShift Container Platform 구성 요소에서 Kubernetes API 서버 및 OpenShift API 서버와 같은 주요 API 그룹의 시작을 관리하는 데 사용됩니다. **openshift.io/run-level** 레이블이 설정된 경우 해당 네임스페이스의 Pod에 SCC가 적용되지 않으므로 해당 네임스페이스에서 실행되는 모든 워크로드에 높은 권한이 부여됩니다.

15.1.1. 기본 보안 컨텍스트 제약 조건

아래 표에 설명된 대로 클러스터에는 몇 가지 기본 SCC(보안 컨텍스트 제약 조건)가 포함되어 있습니다. OpenShift Container Platform에 Operator 또는 기타 구성 요소를 설치할 때 추가 SCC가 설치될 수 있습니다.

중요

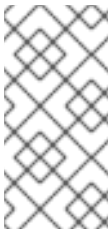
기본 SCC를 수정하지 마십시오. 기본 SCC를 사용자 정의하면 일부 플랫폼 Pod 배포 또는 OpenShift Container Platform이 업그레이드되는 경우 문제가 발생할 수 있습니다. 일부 버전의 OpenShift Container Platform 간에 업그레이드하는 동안 기본 SCC의 값이 기본값으로 재설정되어 해당 SCC에 대한 모든 사용자 정의가 삭제됩니다. 대신 필요에 따라 새 SCC를 만듭니다.

표 15.1. 기본 보안 컨텍스트 제약 조건

보안 컨텍스트 제약 조건	설명
anyuid	**restricted** SCC의 모든 기능을 제공하지만 사용자는 모든 UID 및 GID로 실행할 수 있습니다.
hostaccess	모든 호스트 네임스페이스에 액세스할 수 있지만 네임스페이스에 할당된 UID 및 SELinux 컨텍스트를 사용하여 pod를 실행해야 합니다. **주의** 이 SCC를 사용하면 네임스페이스, 파일 시스템 및 PID에 대한 호스트에 액세스할 수 있습니다. 신뢰할 수 있는 pod에서만 사용해야 합니다. 주의해서 실행합니다.
hostmount-anyuid	**restricted** SCC의 모든 기능을 제공하지만 호스트는 시스템의 모든 UID 및 GID로 마운트하고 실행할 수 있습니다. **주의** 이 SCC를 사용하면 UID 0을 포함하여 모든 UID로 호스트 파일 시스템에 액세스할 수 있습니다. 주의해서 실행합니다.

보안 컨텍스트 제약 조건	설명
hostnetwork	호스트 네트워킹 및 호스트 포트를 사용할 수 있지만 네임스페이스에 할당된 UID 및 SELinux 컨텍스트로 pod를 실행해야 합니다. **주의** 컨트롤 플레인 호스트(마스터 호스트라고도 함)에서 추가 워크로드가 실행되는 경우 **hostnetwork** 에 액세스할 때 주의해야 합니다. 컨트롤 플레인 호스트에서 **hostnetwork**를 실행하는 워크로드는 클러스터에 효과적으로 루팅되므로 신뢰해야 합니다.
node-exporter	Prometheus 노드 내보내기에 사용됩니다. **주의** 이 SCC를 사용하면 UID 0을 포함하여 모든 UID로 호스트 파일 시스템에 액세스할 수 있습니다. 주의해서 실행합니다.
nonroot	**restricted** SCC의 모든 기능을 제공하지만 사용자는 루트 이외의 UID로 실행할 수 있습니다. 사용자는 UID를 지정하거나 컨테이너 런타임의 매니페스트에 지정해야 합니다.

보안 컨텍스트 제약 조건	설명
privileged	모든 권한 및 호스트 기능에 대한 액세스를 허용하고 모든 사용자, 그룹, FSGroup 및 모든 SELinux 컨텍스트로 실행할 수 있습니다. **주의** 이는 가장 완화된 SCC이며 클러스터 관리에만 사용해야 합니다. 주의해서 실행합니다. **권한 있는** SCC를 사용하면 다음을 수행할 수 있습니다. • 사용자가 권한 있는 Pod 실행 • Pod에서 호스트 디렉터리를 볼륨으로 마운트 • Pod를 모든 사용자로 실행 • Pod를 모든 MCS 라벨로 실행 • Pod에서 호스트의 IPC 네임스페이스 사용 • Pod에서 호스트의 PID 네임스페이스 사용 • Pod에서 모든 FSGroup 사용 • Pod에서 추가 그룹 사용 • Pod에서 seccomp 프로필 사용 • Pod에서 모든 기능 요청 **참고** Pod 사양에 **privileged: true** 를 설정하면 권한**있는** SCC가 반드시 선택되지는 않습니다. **PrivilegedContainer: true를 허용하고** 사용자가 사용할 권한이 있는 경우 우선순위가 가장 높은 SCC를 선택합니다.

보안 컨텍스트 제약 조건	설명
restricted	모든 호스트 기능에 대한 액세스를 거부하고 네임스페이스에 할당된 UID 및 SELinux 컨텍스트로 Pod를 실행해야 합니다. 이는 새 설치에서 제공하는 가장 제한적인 SCC 이며 기본적으로 인증된 사용자에게 사용됩니다. **제한된** SCC의 경우 다음을 수행합니다. ● Pod가 권한에 따라 실행되지 않도록 합니다. ● Pod에서 호스트 디렉터리 볼륨을 마운트할 수 없는지 확인합니다. ● 미리 할당된 UID 범위에서 Pod를 사용자로 실행해야 합니다. ● Pod를 사전 할당된 MCS 라벨로 실행해야 합니다. ● Pod에서 FSGroup을 사용하도록 허용 ● Pod에서 추가 그룹을 사용하도록 허용  **참고** restricted SCC는 기본적으로 시스템과 함께 제공되는 SCC의 가장 제한 사항입니다. 그러나 더 제한적인 사용자 지정 SCC를 생성할 수 있습니다. 예를 들어 **readOnlyRootFS** 를 **true** 로 제한하고 **PrivilegeEscalation을 false 로 허용하는** SCC를 생성할 수 있습니다.

15.1.2. 보안 컨텍스트 제약 조건 설정

SCC(보안 컨텍스트 제약 조건)는 pod에서 액세스할 수 있는 보안 기능을 제어하는 설정 및 전략으로 구성됩니다. 이 설정은 세 가지 범주로 분류됩니다.

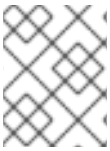
카테고리	설명
부울로 제어	이 유형의 필드는 기본적으로 가장 제한적인 값으로 설정됩니다. 예를 들어, **AllowPrivilegedContainer**는 값이 지정되지 않은 경우 항상**false**로 설정됩니다.
허용 가능한 설정으로 제어	이 유형의 필드는 해당 값이 허용되는지 확인하기 위해 설정과 대조됩니다.
전략으로 제어	가치를 생성하는 전략이 있는 항목에서는 다음을 제공합니다. ● 가치를 생성하는 메커니즘 ● 지정된 값이 허용된 값 집합에 속하도록 하는 메커니즘

CRI-O에는 Pod의 각 컨테이너에 허용되는 다음 기본 기능 목록이 있습니다.

- **CHOWN**
- **DAC_OVERRIDE**

- **FSETID**
- **FOWNER**
- **SETGID**
- **SETUID**
- **SETPCAP**
- **NET_BIND_SERVICE**
- **KILL**

컨테이너에서는 이 기본 목록의 기능을 사용하지만 Pod 매니페스트 작성자는 추가 기능을 요청하거나 일부 기본 동작을 제거하여 목록을 변경할 수 있습니다. **allowedCapabilities**,**defaultAddCapabilities** 및 **requiredDropCapabilities** 매개변수를 사용하여 Pod에서 이러한 요청을 제어합니다. 이러한 매개변수를 사용하여 요청할 수 있는 기능, 각 컨테이너에 추가해야 하는 기능, 각 컨테이너에서 금지 또는 삭제해야 하는 기능을 지정할 수 있습니다.



참고

requiredDropCapabilities 매개변수를 **ALL** 로 설정하여 컨테이너에서 모든 기능을 삭제할 수 있습니다.

15.1.3. 보안 컨텍스트 제약 조건 전략

RunAsUser

- **MustRunAs** - **runAsUser**를 구성해야 합니다. 구성된 **runAsUser**를 기본값으로 사용합니다. 구성된 **runAsUser**에 대해 검증합니다.

  MustRunAs 스니펫 예

```
...
runAsUser:
  type: MustRunAs
  uid: <id>
...
```

- **MustRunAsRange** - 사전 할당된 값을 사용하지 않는 경우 최솟값과 최댓값을 정의해야 합니다. 최솟값을 기본값으로 사용합니다. 전체 허용 범위에 대해 검증합니다.

  MustRunAsRange 스니펫 예

```
...
runAsUser:
  type: MustRunAsRange
  uidRangeMax: <maxvalue>
  uidRangeMin: <minvalue>
...
```

- **MustRunAsNonRoot** - Pod를 0이 아닌 **runAsUser**를 사용하여 제출하거나 Pod의 이미지에 **USER** 지시문이 정의되어 있어야 합니다. 기본값이 제공되지 않습니다.

MustRunAsNonRoot 스니펫 예

```
...
runAsUser:
  type: MustRunAsNonRoot
...
```

- **RunAsAny** - 기본값이 제공되지 않습니다. 모든 **runAsUser**를 지정할 수 있습니다.

RunAsAny 조각의 예

```
...
runAsUser:
  type: RunAsAny
...
```

SELinuxContext

- **MustRunAs** - 사전 할당된 값을 사용하지 않는 경우 **seLinuxOptions**를 구성해야 합니다. **seLinuxOptions**를 기본값으로 사용합니다. **seLinuxOptions**에 대해 검증합니다.
- **RunAsAny** - 기본값이 제공되지 않습니다. 모든 **seLinuxOptions**를 지정할 수 있습니다.

SupplementalGroups

- **MustRunAs** - 사전 할당된 값을 사용하지 않는 경우 범위를 하나 이상 지정해야 합니다. 첫 번째 범위의 최솟값을 기본값으로 사용합니다. 모든 범위에 대해 검증합니다.
- **RunAsAny** - 기본값이 제공되지 않습니다. 임의의 **supplementalGroups**을 지정할 수 있습니다.

FSGroup

- **MustRunAs** - 사전 할당된 값을 사용하지 않는 경우 범위를 하나 이상 지정해야 합니다. 첫 번째 범위의 최솟값을 기본값으로 사용합니다. 첫 번째 범위의 첫 번째 ID에 대해 검증합니다.
- **RunAsAny** - 기본값이 제공되지 않습니다. **fsGroup** ID를 지정할 수 있습니다.

15.1.4. 볼륨 제어

SCC의 **volumes** 필드를 설정하여 특정 볼륨 유형의 사용을 제어할 수 있습니다. 이 필드에 허용되는 값은 볼륨 생성 시 정의되는 볼륨 소스에 해당합니다.

- **awsElasticBlockStore**
- **azureDisk**
- **azureFile**
- **cephFS**
- **Cinder**
- **configMap**

- **downwardAPI**
- **emptyDir**
- **fc**
- **flexVolume**
- **flocker**
- **gcePersistentDisk**
- **gitRepo**
- **glusterfs**
- **hostPath**
- **iscsi**
- **nfs**
- **persistentVolumeClaim**
- **photonPersistentDisk**
- **portworxVolume**
- **projected**
- **quobyte**
- **rbd**
- **scaleIO**
- **secret**
- **storageos**
- **vsphereVolume**
- ***** (모든 볼륨 유형의 사용을 허용하는 특수 값)
- **none** (모든 볼륨 유형의 사용을 허용하지 않는 특수 값입니다. 이전 버전과의 호환성을 위해서만 존재합니다.)

새 SCC에 허용되는 볼륨의 최소 권장 집합은 **configMap**, **downwardAPI**, **emptyDir**, **persistentVolumeClaim**, **secret**, **projected**입니다.

참고

OpenShift Container Platform의 각 릴리스에 새로운 유형이 추가되므로 허용되는 볼륨 유형 목록은 포괄적이지 않습니다.

참고

이전 버전과의 호환성을 위해 **allowHostDirVolumePlugin**을 사용하면 **volumes** 필드의 설정을 덮어씁니다. 예를 들어, **allowHostDirVolumePlugin**이 false로 설정되어 있지만 **volumes** 필드에서 허용되는 경우 **volumes**에서 **hostPath** 값이 제거됩니다.

15.1.5. 승인 제어

SCC를 통한 *허용 제어*를 사용하면 사용자에게 부여된 기능을 기반으로 리소스 생성을 제어할 수 있습니다.

SCC 측면에서는 허용 컨트롤러가 컨텍스트에서 사용 가능한 사용자 정보를 검사하여 적절한 SCC 집합을 검색할 수 있음을 의미합니다. 이 경우 Pod에 운영 환경에 대한 요청을 하거나 Pod에 적용할 일련의 제약 조건을 생성할 수 있는 권한이 부여됩니다.

허용 작업에서 Pod를 승인하는 데 사용하는 SCC 집합은 사용자 ID 및 사용자가 속하는 그룹에 따라 결정됩니다. 또한 Pod에서 서비스 계정을 지정하는 경우, 허용된 SCC 집합에 서비스 계정에 액세스할 수 있는 모든 제약 조건이 포함됩니다.

허용 작업에서는 다음 방법을 사용하여 Pod에 대한 최종 보안 컨텍스트를 생성합니다.

1. 사용 가능한 모든 SCC를 검색합니다.
2. 요청에 지정되지 않은 보안 컨텍스트 설정에 대한 필드 값을 생성합니다.
3. 사용 가능한 제약 조건에 대해 최종 설정을 검증합니다.

일치하는 제약 조건 집합이 있는 경우 Pod가 승인됩니다. 요청을 SCC와 일치시킬 수 없는 경우 Pod가 거부됩니다.

Pod는 SCC에 대해 모든 필드를 검증해야 합니다. 다음은 검증이 필요한 필드 중 단 2개의 예입니다.

참고

이러한 예제는 사전 할당된 값을 사용하는 전략 컨텍스트에 있습니다.

MustRunAs의 FSGroup SCC 전략

Pod에서 **fsGroup** ID를 정의하는 경우 해당 ID는 기본 **fsGroup** ID와 같아야 합니다. 그렇지 않으면 해당 SCC에서 Pod를 검증하지 않고 다음 SCC를 평가합니다.

SecurityContextConstraints.fsGroup 필드에 값 **RunAsAny**가 있고 Pod 사양에서 **Pod.spec.securityContext.fsGroup**을 생략하는 경우, 이 필드는 유효한 것으로 간주됩니다. 유효성을 확인하는 동안 다른 SCC 설정에서 다른 Pod 필드를 거부하여 Pod가 실패할 수 있습니다.

MustRunAs의 SupplementalGroups SCC 전략

Pod 사양에서 하나 이상의 **supplementalGroups** ID를 정의하는 경우, Pod의 ID는 네임스페이스의 **openshift.io/sa.scc.supplemental-groups** 주석에 있는 ID 중 하나와 같아야 합니다. 그렇지 않으면 해당 SCC에서 Pod를 검증하지 않고 다음 SCC를 평가합니다.

SecurityContextConstraints.supplementalGroups 필드에 값 **RunAsAny**가 있고 Pod 사양에서 **Pod.spec.securityContext.supplementalGroups**를 생략하는 경우, 이 필드는 유효한 것으로 간주됩니다. 유효성을 확인하는 동안 다른 SCC 설정에서 다른 Pod 필드를 거부하여 Pod가 실패할 수 있습니다.

15.1.6. 보안 컨텍스트 제약 조건 우선순위 지정

SCC(보안 컨텍스트 제약 조건)에는 승인 컨트롤러의 요청을 확인할 때 순서에 영향을 주는 우선순위 필드가 있습니다.

우선순위 **0** 은 가능한 가장 낮은 우선 순위입니다. nil 우선순위는 **0** 또는 가장 낮은 우선순위로 간주됩니다. 정렬 시 우선순위가 높은 SCC가 집합의 앞쪽으로 이동합니다.

사용 가능한 전체 SCC 집합이 결정되면 SCC가 다음과 같은 방식으로 정렬됩니다.

1. 우선순위가 가장 높은 SCC가 먼저 정렬됩니다.
2. 우선순위가 동일한 경우 SCC는 가장 제한적인 것에서 최소 제한으로 정렬됩니다.
3. 우선순위와 제한이 모두 동일한 경우 SCC는 이름별로 정렬됩니다.

기본적으로 클러스터 관리자에게 부여되는 **anyuid** SCC는 SCC 집합에서 우선순위가 부여됩니다. 이를 통해 클러스터 관리자는 Pod의 **SecurityContext** 에 **RunAsUser** 를 지정하여 모든 사용자로 Pod를 실행할 수 있습니다.

15.2. 미리 할당된 보안 컨텍스트 제약 조건 값 정보

허용 컨트롤러는 SCC(보안 컨텍스트 제약 조건)의 특정 조건을 인식하여 네임스페이스에서 미리 할당된 값을 조회하고 pod를 처리하기 전에 SCC를 채울 수 있습니다. 각 SCC 전략은 실행 중인 pod에 정의된 다양한 ID에 대한 최종 값을 만들기 위해 pod 사양 값으로 집계된 각 정책에 대해, 허용된 경우 사전 할당된 값을 사용하여 다른 전략과 독립적으로 평가됩니다.

다음 SCC를 사용하면 Pod 사양에 범위가 정의되지 않은 경우 허용 컨트롤러에서 사전 할당된 값을 찾습니다.

1. 최소 또는 최대 집합이 없는 **MustRunAsRange**의 **RunAsUser** 전략. 허용 작업에서는 범위 필드를 채우기 위해 **openshift.io/sa.scc.uid-range** 주석을 찾습니다.
2. 수준이 설정되지 않은 **MustRunAs**의 **SELinuxContext** 전략. 허용 작업에서는 수준을 채울 **openshift.io/sa.scc.mcs** 주석을 찾습니다.
3. **MustRunAs**의 **FSGroup** 전략. 허용 작업에서는 **openshift.io/sa.scc.supplemental-group** 주석을 찾습니다.
4. **MustRunAs**의 **SupplementalGroups** 전략. 허용 작업에서는 **openshift.io/sa.scc.supplemental-group** 주석을 찾습니다.

생성 단계 중 보안 컨텍스트 공급자는 Pod에서 구체적으로 설정되지 않은 모든 매개변수 값으로 기본값을 사용합니다. 기본값은 선택한 전략에 따라 다릅니다.

1. **RunAsAny** 및 **MustRunAsNonRoot** 전략에서는 기본값을 제공하지 않습니다. Pod에 그룹 ID와 같은 매개변수 값이 필요한 경우, Pod 사양에 값을 정의해야 합니다.
2. **MustRunAs** (단일 값) 전략에서는 항상 사용되는 기본값을 제공합니다. 예를 들어, 그룹 ID의 경우 Pod 사양에서 고유한 ID 값을 정의하더라도 네임스페이스의 기본 매개변수 값도 Pod의 그룹에 나타납니다.
3. **MustRunAsRange** 및 **MustRunAs** (범위 기반) 전략에서는 최소 범위 값을 제공합니다. 단일 값 **MustRunAs** 전략과 마찬가지로 네임스페이스의 기본 매개 변수 값이 실행 중인 Pod에 나타납니다. 범위 기반 전략을 여러 범위로 구성할 수 있는 경우, 처음 구성된 최소 범위 값을 제공합니다.

참고

openshift.io/sa.scc.supplemental-groups 주석이 네임스페이스에 존재하지 않는 경우, **FSGroup** 및 **SupplementalGroups** 전략이 **openshift.io/sa.scc.uid-range** 주석으로 변경됩니다. 둘 다 존재하지 않으면 SCC가 생성되지 않습니다.

참고

기본적으로 주석 기반 **FSGroup** 전략은 주석의 최솟값에 따라 단일 범위로 자체 구성됩니다. 예를 들어, 주석이 **1/3**이면 **FSGroup** 전략은 최솟값이자 최댓값인 **1**로 구성됩니다. **FSGroup** 필드에 더 많은 그룹을 허용하려면 주석을 사용하지 않는 사용자 정의 SCC를 구성하면 됩니다.

참고

openshift.io/sa.scc.supplemental-groups 주석에는 **<start>/<length** 또는 **<start>-<end>** 형식의 쉼표로 구분된 블록 목록을 사용할 수 있습니다. **openshift.io/sa.scc.uid-range** 주석에는 단일 블록만 사용할 수 있습니다.

15.3. 보안 컨텍스트 제약 조건의 예

다음 예에서는 SCC(보안 컨텍스트 제약 조건)의 형식 및 주석을 보여줍니다.

주석이 달린 권한 있는 SCC

```
allowHostDirVolumePlugin: true
allowHostIPC: true
allowHostNetwork: true
allowHostPID: true
allowHostPorts: true
allowPrivilegedContainer: true
allowedCapabilities: 1
- '*'
apiVersion: security.openshift.io/v1
defaultAddCapabilities: [] 2
fsGroup: 3
  type: RunAsAny
groups: 4
- system:cluster-admins
- system:nodes
kind: SecurityContextConstraints
metadata:
  annotations:
    kubernetes.io/description: 'privileged allows access to all privileged and host
      features and the ability to run as any user, any group, any fsGroup, and with
      any SELinux context.  WARNING: this is the most relaxed SCC and should be used
      only for cluster administration. Grant with caution.'
  creationTimestamp: null
  name: privileged
priority: null
readOnlyRootFilesystem: false
requiredDropCapabilities: 5
- KILL
- MKNOD
```

```
- SETUID
- SETGID
runAsUser: 6
  type: RunAsAny
seLinuxContext: 7
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups: 8
  type: RunAsAny
users: 9
- system:serviceaccount:default:registry
- system:serviceaccount:default:router
- system:serviceaccount:openshift-infra:build-controller
volumes:
- '*'
```

1. pod에서 요청할 수 있는 기능 목록입니다. 빈 목록은 기능을 요청할 수 없음을 나타내고, 특수 기호 *는 모든 기능을 요청할 수 있음을 나타냅니다.
2. 임의의 pod에 추가된 추가 기능 목록입니다.
3. 보안 컨텍스트에 허용되는 값을 지시하는 **FSGroup** 전략입니다.
4. 이 SCC에 액세스할 수 있는 그룹입니다.
5. Pod에서 삭제할 기능 목록입니다. 또는 모든 기능을 **삭제하려면 ALL** 을 지정합니다.
6. 보안 컨텍스트에 허용되는 값을 지시하는 **runAsUser** 전략 유형입니다.
7. 보안 컨텍스트에 허용되는 값을 지시하는 **seLinuxContext** 전략 유형입니다.
8. **supplementalGroups** 전략으로, 보안 컨텍스트에 허용되는 추가 그룹을 나타냅니다.
9. 이 SCC에 액세스할 수 있는 사용자입니다.

SCC의 **users** 및 **groups** 필드는 SCC에 액세스할 수 있는 사용자를 제어합니다. 기본적으로 클러스터 관리자, 노드 및 빌드 컨트롤러에는 권한 있는 SCC에 대한 액세스 권한이 부여됩니다. 인증된 모든 사용자에게는 제한된 SCC에 대한 액세스 권한이 부여됩니다.

명시적인 runAsUser 설정이 없는 경우

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext: 1
  containers:
  - name: sec-ctx-demo
    image: gcr.io/google-samples/node-hello:1.0
```

1 컨테이너 또는 pod에서 실행해야 하는 사용자 ID를 요청하지 않는 경우, 유효 UID는 이 pod를 내보내는 SCC에 따라 다릅니다. 제한된 SCC는 기본적으로 인증된 모든 사용자에게 부여되므로 모든 사용자 및 서비스 계정에서 사용할 수 있으며, 대부분의 사례에서 사용됩니다. 제한된 SCC는 **securityContext.runAsUser** 필드의 사용 가능한 값을 제한하고 기본값을 설정하는 데 **MustRunAsRange** 전략을 사용합니다. 허용 플러그인은 이 범위를 제공하지 않기 때문에 현재 프로젝트에서 **openshift.io/sa.scc.uid-range** 주석을 찾아 범위 필드를 채웁니다. 결국 컨테이너에는 모든 프로젝트의 범위가 다르기 때문에 예측하기 어려운 범위의 첫 번째 값과 동일한 **runAsUser**가 있게 됩니다.

명시적인 runAsUser 설정

```
apiVersion: v1
kind: Pod
metadata:
  name: security-context-demo
spec:
  securityContext:
    runAsUser: 1000 1
  containers:
    - name: sec-ctx-demo
      image: gcr.io/google-samples/node-hello:1.0
```

1 특정 사용자 ID를 요청하는 컨테이너 또는 Pod는 서비스 계정 또는 사용자에게 해당 사용자 ID를 허용하는 SCC에 대한 액세스 권한이 부여된 경우에만 OpenShift Container Platform에서 승인됩니다. SCC를 사용하면 임의의 ID, 특정 범위에 속하는 ID 또는 요청과 관련된 정확한 사용자 ID를 허용할 수 있습니다.

이 구성은 SELinux, fsGroup 및 추가 그룹에 유효합니다.

15.4. 보안 컨텍스트 제약 조건 생성

OpenShift CLI(**oc**)를 사용하여 SCC(보안 컨텍스트 제약 조건)를 생성할 수 있습니다.

사전 요구 사항

- OpenShift CLI(**oc**)를 설치합니다.
- **cluster-admin** 역할의 사용자로 클러스터에 로그인합니다.

절차

1. 이름이 **scc_admin.yaml**인 YAML 파일에 SCC를 정의합니다.

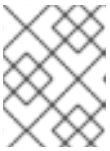
   SecurityContextConstraints 오브젝트 정의

   ```
   kind: SecurityContextConstraints
   apiVersion: security.openshift.io/v1
   metadata:
     name: scc-admin
   allowPrivilegedContainer: true
   runAsUser:
     type: RunAsAny
   seLinuxContext:
   ```

```
  type: RunAsAny
fsGroup:
  type: RunAsAny
supplementalGroups:
  type: RunAsAny
users:
- my-admin-user
groups:
- my-admin-group
```

필요한 경우 **requiredDropCapabilities** 필드를 원하는 값으로 설정하여 SCC의 특정 기능을 삭제할 수 있습니다. 지정된 기능은 컨테이너에서 삭제됩니다. 모든 기능을 삭제하려면 all을 **지정합니다**. 예를 들어 **KILL,MKNOD** 및 **SYS_CHROOT** 기능을 삭제하는 SCC를 생성하려면 SCC 오브젝트에 다음을 추가합니다.

```
requiredDropCapabilities:
- KILL
- MKNOD
- SYS_CHROOT
```



참고

allowedCapabilities 및 requiredDropCapabilities 의 기능은 나열할 수 없습니다.

CRI-O는 Docker 문서에 있는 동일한 기능 값 목록을 지원합니다.

2. 파일에서 전달하여 SCC 생성:

```
$ oc create -f scc_admin.yaml
```

출력 예

```
securitycontextconstraints "scc-admin" created
```

검증

- SCC가 생성되었는지 확인합니다.

```
$ oc get scc scc-admin
```

출력 예

```
NAME      PRIV    CAPS    SELINUX    RUNASUSER    FSGROUP    SUPGROUP
PRIORITY    READONLYROOTFS    VOLUMES
scc-admin    true    []    RunAsAny    RunAsAny    RunAsAny    RunAsAny    <none>    false
[awsElasticBlockStore azureDisk azureFile cephFS cinder configMap downwardAPI
emptyDir fc flexVolume flocker gcePersistentDisk gitRepo glusterfs iscsi nfs
persistentVolumeClaim photonPersistentDisk quobyte rbd secret vsphere]
```

15.5. 보안 컨텍스트 제약 조건에 대한 역할 기반 액세스

SBA를 RBAC에서 처리하는 리소스로 지정할 수 있습니다. 그러면 SCC에 대한 액세스 권한의 범위를 특정 프로젝트 또는 전체 클러스터로 지정할 수 있습니다. SCC에 사용자, 그룹 또는 서비스 계정을 직접 할당하면 클러스터 전체 범위가 유지됩니다.

참고

기본 네임스페이스(**default**, **kube-system**, **kube-public**, **openshift-node**, **openshift-infra**, **openshift**) 중 하나에서 생성된 Pod에는 SCC를 할당할 수 없습니다. 이러한 네임스페이스는 Pod 또는 서비스를 실행하는 데 사용해서는 안 됩니다.

역할에 대한 SCC 액세스 권한을 포함하려면 역할을 생성할 때 **scc** 리소스를 지정하십시오.

```
$ oc create role <role-name> --verb=use --resource=scc --resource-name=<scc-name> -n
<namespace>
```

그러면 다음과 같은 역할 정의가 생성됩니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
...
  name: role-name (1)
  namespace: namespace (2)
...
rules:
- apiGroups:
  - security.openshift.io (3)
  resourceNames:
  - scc-name (4)
  resources:
  - securitycontextconstraints (5)
  verbs: (6)
  - use
```

(1) 역할의 이름입니다.

(2) 정의된 역할의 네임스페이스입니다. 지정하지 않는 경우 기본값은 **default**입니다.

(3) **SecurityContextConstraints** 리소스를 포함하는 API 그룹입니다. **scc**가 리소스로 지정되면 자동으로 정의됩니다.

(4) 액세스할 SCC 이름의 예입니다.

(5) 사용자가 **resourceNames** 필드에 SCC 이름을 지정할 수 있는 리소스 그룹의 이름입니다.

(6) 역할에 적용할 동사 목록입니다.

이러한 규칙이 있는 로컬 또는 클러스터 역할을 통해 RoleBinding 또는 ClusterRoleBinding으로 바인딩된 주체는 **scc-name**이라는 사용자 정의 SCC를 사용할 수 있습니다.

참고

RBAC는 에스컬레이션되지 않도록 설계되었으므로 프로젝트 관리자도 SCC에 대한 액세스 권한을 부여할 수 없습니다. **restricted** SCC를 포함하여 기본적으로 SCC 리소스에 동사 **use**를 사용할 수 없습니다.

15.6. 보안 컨텍스트 제약 조건 명령 참조

OpenShift CLI (**oc**)를 사용하여 인스턴스의 SCC(보안 컨텍스트 제약 조건)를 일반 API 오브젝트로 관리할 수 있습니다.

참고

SCC를 관리하려면 **cluster-admin** 권한이 있어야 합니다.

15.6.1. 보안 컨텍스트 제약 조건 나열

현재 SCC 목록을 가져오려면 다음을 실행합니다.

```
$ oc get scc
```

출력 예

```
NAME              PRIV    CAPS    SELINUX     RUNASUSER        FSGROUP     SUPGROUP
PRIORITY   READONLYROOTFS   VOLUMES
anyuid            false   []      MustRunAs   RunAsAny         RunAsAny    RunAsAny    10
false      [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
hostaccess        false   []      MustRunAs   MustRunAsRange   MustRunAs   RunAsAny    <none>
false      [configMap downwardAPI emptyDir hostPath persistentVolumeClaim projected secret]
hostmount-anyuid  false   []      MustRunAs   RunAsAny         RunAsAny    RunAsAny    <none>
false      [configMap downwardAPI emptyDir hostPath nfs persistentVolumeClaim projected
secret]
hostnetwork       false   []      MustRunAs   MustRunAsRange   MustRunAs   MustRunAs   <none>
false      [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
node-exporter     false   []      RunAsAny    RunAsAny         RunAsAny    RunAsAny    <none>   false
[*]
nonroot           false   []      MustRunAs   MustRunAsNonRoot RunAsAny    RunAsAny    <none>
false      [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
privileged        true    [*]     RunAsAny    RunAsAny         RunAsAny    RunAsAny    <none>   false
[*]
restricted        false   []      MustRunAs   MustRunAsRange   MustRunAs   RunAsAny    <none>
false      [configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
```

15.6.2. 보안 컨텍스트 제약 조건 검사

SCC가 적용되는 사용자, 서비스 계정 및 그룹을 포함하여 특정 SCC에 대한 정보를 볼 수 있습니다.

예를 들어 **restricted** SCC를 검사하려면 다음을 실행합니다.

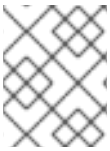
```
$ oc describe scc restricted
```

출력 예

```
Name:     restricted
Priority:   <none>
Access:
  Users:    <none> (1)
  Groups:    system:authenticated (2)
Settings:
  Allow Privileged:   false
  Default Add Capabilities:  <none>
  Required Drop Capabilities:  KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities:   <none>
  Allowed Seccomp Profiles:  <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,secret
  Allow Host Network:   false
  Allow Host Ports:   false
  Allow Host PID:   false
  Allow Host IPC:   false
  Read Only Root Filesystem:  false
  Run As User Strategy: MustRunAsRange
    UID:    <none>
    UID Range Min:   <none>
    UID Range Max:   <none>
  SELinux Context Strategy: MustRunAs
    User:    <none>
    Role:    <none>
    Type:    <none>
    Level:    <none>
  FSGroup Strategy: MustRunAs
    Ranges:    <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges:    <none>
```

(1) SCC가 적용되는 사용자 및 서비스 계정을 나열합니다.

(2) SCC가 적용되는 그룹을 나열합니다.



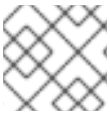
참고

업그레이드하는 동안 사용자 정의 SCC를 유지하려면 기본 SCC의 설정을 편집하지 마십시오.

15.6.3. 보안 컨텍스트 제약 조건 삭제

SCC를 삭제하려면 다음을 수행합니다.

```
$ oc delete scc <scc_name>
```



참고

기본 SCC를 삭제하면 클러스터 재시작 시 기본 SCC가 다시 생성됩니다.

15.6.4. 보안 컨텍스트 제약 조건 업데이트

기존 SCC를 업데이트하려면 다음을 수행합니다.

```
$ oc edit scc <scc_name>
```

참고

업그레이드하는 동안 사용자 정의 SCC를 유지하려면 기본 SCC의 설정을 편집하지 마십시오.

16장. SYSTEM:ADMIN 사용자 가장

16.1. API 가장

OpenShift Container Platform API에 대한 요청을 다른 사용자가 보낸 것처럼 작동하도록 구성할 수 있습니다. 자세한 내용은 Kubernetes 설명서의 사용자 가장을 참조하십시오.

16.2. SYSTEM:ADMIN 사용자 가장

사용자에게 **system:admin**을 가장할 수 있는 권한을 부여하여 클러스터 관리자 권한을 부여할 수 있습니다.

절차

- 사용자에게 **system:admin**을 가장할 수 있는 권한을 부여하려면 다음 명령을 실행합니다.

```
$ oc create clusterrolebinding <any_valid_name> --clusterrole=sudoer --user=<username>
```

작은 정보

또는 다음 YAML을 적용하여 **system:admin**을 가장할 수 있는 권한을 부여할 수 있습니다.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <any_valid_name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: sudoer
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: <username>
```

16.3. SYSTEM:ADMIN 그룹 가장

system:admin 사용자에게 그룹을 통해 클러스터 관리 권한을 부여하는 경우, 연결된 그룹을 가장하려면 명령에 **--as=<user> --as-group=<group1> --as-group=<group2>** 매개변수를 포함해야 합니다.

절차

- 연결된 클러스터 관리 그룹을 가장하여 **system:admin**을 가장할 수 있는 사용자 권한을 부여하려면 다음 명령을 실행합니다.

```
$ oc create clusterrolebinding <any_valid_name> --clusterrole=sudoer --as=<user> \
--as-group=<group1> --as-group=<group2>
```

17장. LDAP 그룹 동기화

관리자는 그룹을 사용하여 사용자를 관리하고 권한을 변경하며 협업을 개선할 수 있습니다. 조직에서 이미 사용자 그룹을 생성하여 LDAP 서버에 저장했을 수 있습니다. OpenShift Container Platform은 이러한 LDAP 레코드를 내부 OpenShift Container Platform 레코드와 동기화하므로 한 곳에서 그룹을 관리할 수 있습니다. OpenShift Container Platform은 현재 그룹 멤버십을 정의하는 세 가지 공통 스키마를 사용하여 LDAP 서버와 그룹 동기화를 지원합니다. RFC 2307, Active Directory 및 보강된 Active Directory.

LDAP 구성에 대한 자세한 내용은 LDAP ID 공급자 구성을 참조하십시오.

참고

그룹을 동기화하려면 **cluster-admin** 권한이 있어야 합니다.

17.1. LDAP 동기화 구성 정보

LDAP 동기화를 실행하려면 동기화 구성 파일이 필요합니다. 이 파일에는 다음과 같은 LDAP 클라이언트 구성 세부 사항이 있습니다.

- LDAP 서버 연결에 필요한 구성
- LDAP 서버에서 사용되는 스키마에 종속된 동기화 구성 옵션
- OpenShift Container Platform 그룹 이름을 LDAP 서버의 그룹에 매핑하는 관리자 정의 이름 매핑 목록

구성 파일의 형식은 사용 중인 스키마에 따라 다릅니다. RFC 2307, Active Directory 또는 보강된 Active Directory.

LDAP 클라이언트 구성

구성의 LDAP 클라이언트 구성 섹션에서는 LDAP 서버에 대한 연결을 정의합니다.

구성의 LDAP 클라이언트 구성 섹션에서는 LDAP 서버에 대한 연결을 정의합니다.

LDAP 클라이언트 구성

```
url: ldap://10.0.0.0:389 1
bindDN: cn=admin,dc=example,dc=com 2
bindPassword: password 3
insecure: false 4
ca: my-ldap-ca-bundle.crt 5
```

1. 연결 프로토콜, 데이터베이스를 호스팅하는 LDAP 서버의 IP 주소, 연결할 포트로 **scheme://host:port** 형식으로 되어 있습니다.
2. 바인딩 DN으로 사용할 선택적 고유 이름(DN)입니다. OpenShift Container Platform에서는 동기화 작업을 위한 항목을 검색하는 데 높은 권한이 필요한 경우 이 고유 이름을 사용합니다.
3. 바인딩에 사용할 선택적 암호입니다. OpenShift Container Platform에서는 동기화 작업을 위한 항목을 검색하는 데 높은 권한이 필요한 경우 이 암호를 사용합니다. 이 값은 환경 변수, 외부 파일 또는 암호화된 파일로 제공될 수도 있습니다.
4.

false인 경우 보안 LDAP(**ldaps://**) URL이 TLS를 사용하여 연결되고 안전하지 않은 LDAP(**ldap://**) URL은 TLS로 업그레이드됩니다. **true**인 경우 서버에 TLS 연결이 이루어지지 않으며 **ldaps://** URL

5 구성된 URL에 대한 서버 인증서의 유효성을 확인하는 데 사용할 인증서 번들입니다. 비어있는 경우 OpenShift Container Platform은 시스템에서 신뢰하는 루트를 사용합니다. **insecure**가 **false**로 설정된 경우에만 적용됩니다.

LDAP 쿼리 정의

동기화 구성은 동기화에 필요한 항목에 대한 LDAP 쿼리 정의로 구성됩니다. LDAP 쿼리의 특정 정의는 LDAP 서버에 멤버십 정보를 저장하는 데 사용하는 스키마에 따라 다릅니다.

LDAP 쿼리 정의

```
baseDN: ou=users,dc=example,dc=com 1
scope: sub 2
derefAliases: never 3
timeout: 0 4
filter: (objectClass=person) 5
pageSize: 0 6
```

1 모든 검색이 시작되는 디렉터리 분기의 DN(고유 이름)입니다. 디렉터리 트리의 최상위를 지정해야 하지만 디렉터리의 하위 트리를 지정할 수도 있습니다.

2 검색 범위입니다. 유효한 값은 **base**, **one** 또는 **sub**입니다. 값이 정의되지 않은 경우 **sub** 범위로 가정됩니다. 범위 옵션에 대한 설명은 아래 표를 참조하십시오.

3 LDAP 트리의 별칭에 관한 검색 동작입니다. 유효한 값은 **never**, **search**, **base** 또는 **always**입니다. 값이 정의되지 않은 경우 기본값은 **always** 역참조 별칭으로 설정됩니다. 역참조 동작에 대한 설명은 아래 표를 참조하십시오.

4 클라이언트에서 검색할 수 있는 시간 제한(초)입니다. 값이 **0**이면 클라이언트 쪽 제한이 적용되지 않습니다.

5 유효한 LDAP 검색 필터입니다. 정의되지 않은 경우 기본값은 **(objectClass=*)**입니다.

6 서버 응답 페이지의 최대 크기 옵션으로, LDAP 항목으로 측정합니다. **0**으로 설정하면 응답 페이지에 크기 제한이 없습니다. 클라이언트 또는 서버에서 기본적으로 허용하는 것보다 쿼리에서 더 많은 항목을 반환하는 경우, 페이징 크기를 설정해야 합니다.

표 17.1. LDAP 검색 범위 옵션

LDAP 검색 범위	설명
base	쿼리의 기본 DN에 의해 지정된 오브젝트만 고려합니다.
one	쿼리의 기본 DN과 동일한 수준의 트리에 있는 모든 오브젝트를 고려합니다.
sub	쿼리에 대해 제공된 기본 DN을 기반으로 하는 전체 하위 트리를 고려합니다.

표 17.2. LDAP 역참조 동작

역참조 행동	설명
never	LDAP 트리에 있는 별칭을 역참조하지 않도록 합니다.
search	검색하는 동안 역참조 별칭만 발견되었습니다.
base	기본 오브젝트를 찾는 동안 별칭만 역참조합니다.
always	항상 LDAP 트리에 있는 모든 별칭을 역참조합니다.

사용자 정의 이름 매핑

사용자 정의 이름 매핑은 OpenShift Container Platform 그룹의 이름을 LDAP 서버에서 그룹을 식별하는 고유 식별자에 명시적으로 매핑합니다. 매핑에서는 일반 YAML 구문을 사용합니다. 사용자 정의 매핑은 LDAP 서버의 모든 그룹에 대한 항목을 포함하거나 그룹의 하위 집합만 포함할 수 있습니다. LDAP 서버에 사용자 정의 이름 매핑이 없는 그룹이 있는 경우, 동기화 중 기본 동작은 OpenShift Container Platform 그룹 이름으로 지정된 속성을 사용하는 것입니다.

사용자 정의 이름 매핑

```
groupUIDNameMapping:
  "cn=group1,ou=groups,dc=example,dc=com": firstgroup
  "cn=group2,ou=groups,dc=example,dc=com": secondgroup
  "cn=group3,ou=groups,dc=example,dc=com": thirdgroup
```

17.1.1. RFC 2307 구성 파일 정보

RFC 2307 스키마를 사용하려면 사용자 항목 및 그룹 항목에 대한 LDAP 쿼리 정의와 내부 OpenShift Container Platform 레코드에서 해당 항목을 표시하는 데 사용할 속성을 제공해야 합니다.

명확히 하기 위해, OpenShift Container Platform에서 생성한 그룹의 경우 사용자용 필드 또는 관리자용 필드에 가급적 고유 이름 이외의 속성을 사용해야 합니다. 예를 들면 이메일로 OpenShift Container Platform 그룹의 사용자를 구분하고, 그룹 이름을 공통 이름으로 사용합니다. 다음 구성 파일에서는 이러한 관계를 생성합니다.

참고

사용자 정의 이름 매핑을 사용하는 경우에는 구성 파일이 다릅니다.

RFC 2307 스키마를 사용하는 LDAP 동기화 구성: rfc2307_config.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389 (1)
insecure: false (2)
rfc2307:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
        pageSize: 0
```

```
  groupUIDAttribute: dn 3
  groupNameAttributes: [ cn ] 4
  groupMembershipAttributes: [ member ] 5
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  userUIDAttribute: dn 6
  userNameAttributes: [ mail ] 7
  tolerateMemberNotFoundErrors: false
  tolerateMemberOutOfScopeErrors: false
```

1. 이 그룹의 레코드가 저장된 LDAP 서버의 IP 주소 및 호스트입니다.
2. **false**인 경우 보안 LDAP(**ldaps://**) URL이 TLS를 사용하여 연결되고 안전하지 않은 LDAP(**ldap://**) URL은 TLS로 업그레이드됩니다. **true**인 경우 서버에 TLS 연결이 이루어지지 않으며 **ldaps://** URL 스키마를 사용할 수 없습니다.
3. LDAP 서버에서 그룹을 고유하게 식별하는 속성입니다. **groupUIDAttribute**로 DN을 사용할 때는 **groupsQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.
4. 그룹 이름으로 사용할 속성입니다.
5. 멤버십 정보를 저장하는 그룹의 속성입니다.
6. LDAP 서버에서 사용자를 고유하게 식별하는 속성입니다. userUIDAttribute로 DN을 사용할 때는 **usersQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.
7. OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다.

17.1.2. Active Directory 구성 파일 정보

Active Directory 스키마를 사용하려면 사용자 항목에 대한 LDAP 쿼리 정의와 내부 OpenShift Container Platform 그룹 레코드에서 해당 항목을 표시하는 속성을 제공해야 합니다.

명확히 하기 위해, OpenShift Container Platform에서 생성한 그룹의 경우 사용자용 필드 또는 관리자용 필드에 가급적 고유 이름 이외의 속성을 사용해야 합니다. 예를 들면 OpenShift Container Platform 그룹의 사용자는 이메일로 구분하지만, 그룹 이름은 LDAP 서버의 그룹 이름에 따라 정의합니다. 다음 구성 파일에서는 이러한 관계를 생성합니다.

Active Directory 스키마를 사용하는 LDAP 동기화 구성: active_directory_config.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
activeDirectory:
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=person)
```

```
    pageSize: 0
  userNameAttributes: [ mail ] 1
  groupMembershipAttributes: [ memberOf ] 2
```

1 OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다.

2 멤버십 정보를 저장하는 사용자 속성입니다.

17.1.3. 보강된 Active Directory 구성 파일 정보

보강된 Active Directory 스키마를 사용하려면 사용자 항목 및 그룹 항목에 대한 LDAP 쿼리 정의와 내부 OpenShift Container Platform 그룹 레코드에서 해당 항목을 표시하는 속성을 제공해야 합니다.

명확히 하기 위해, OpenShift Container Platform에서 생성한 그룹의 경우 사용자용 필드 또는 관리자용 필드에 가급적 고유 이름 이외의 속성을 사용해야 합니다. 예를 들면 이메일로 OpenShift Container Platform 그룹의 사용자를 구분하고, 그룹 이름을 공통 이름으로 사용합니다. 다음 구성 파일에서는 이러한 관계를 생성합니다.

보강된 Active Directory 스키마를 사용하는 LDAP 동기화 구성: augmented_active_directory_config.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
augmentedActiveDirectory:
  groupsQuery:
    baseDN: "ou=groups,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  groupUIDAttribute: dn 1
  groupNameAttributes: [ cn ] 2
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=person)
    pageSize: 0
  userNameAttributes: [ mail ] 3
  groupMembershipAttributes: [ memberOf ] 4
```

1 LDAP 서버에서 그룹을 고유하게 식별하는 속성입니다. groupUIDAttribute로 DN을 사용할 때는 **groupsQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.

2 그룹 이름으로 사용할 속성입니다.

3 OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다.

4 멤버십 정보를 저장하는 사용자 속성입니다.

17.2. LDAP 동기화 실행

동기화 구성 파일을 생성하면 동기화를 시작할 수 있습니다. 관리자는 OpenShift Container Platform을 통해 동일한 서버에서 다양한 동기화 유형을 수행할 수 있습니다.

17.2.1. OpenShift Container Platform과 LDAP 서버 동기화

LDAP 서버의 모든 그룹을 OpenShift Container Platform과 동기화할 수 있습니다.

사전 요구 사항

- 동기화 구성 파일을 생성합니다.

절차

- LDAP 서버의 모든 그룹을 OpenShift Container Platform과 동기화하려면 다음을 실행합니다.

```
$ oc adm groups sync --sync-config=config.yaml --confirm
```

참고

기본적으로 모든 그룹 동기화 작업은 시험 실행이므로 OpenShift Container Platform 그룹 레코드를 변경하려면 **oc adm groups sync** 명령에 **--confirm** 플래그를 설정해야 합니다.

17.2.2. LDAP 서버와 OpenShift Container Platform 그룹 동기화

OpenShift Container Platform에 이미 있으며 구성 파일에 지정된 LDAP 서버의 그룹에 해당하는 모든 그룹을 동기화할 수 있습니다.

사전 요구 사항

- 동기화 구성 파일을 생성합니다.

절차

- LDAP 서버와 OpenShift Container Platform 그룹을 동기화하려면 다음을 실행합니다.

```
$ oc adm groups sync --type=openshift --sync-config=config.yaml --confirm
```

참고

기본적으로 모든 그룹 동기화 작업은 시험 실행이므로 OpenShift Container Platform 그룹 레코드를 변경하려면 **oc adm groups sync** 명령에 **--confirm** 플래그를 설정해야 합니다.

17.2.3. OpenShift Container Platform과 LDAP 서버의 하위 그룹 동기화

허용 목록 파일, 차단 목록 파일 또는 둘 다를 사용하여 OpenShift Container Platform과 LDAP 그룹의 하위 집합을 동기화할 수 있습니다.

참고

차단 목록 파일, 허용 목록 파일 또는 허용 목록 리터럴을 조합하여 사용할 수 있습니다. 허용 목록 및 차단 목록 파일에는 한 줄에 하나의 고유한 그룹 식별자를 포함해야 하며, 명령 자체에 허용 목록 리터럴을 직접 포함할 수 있습니다. 이러한 지침은 OpenShift Container Platform에 이미 존재하는 그룹뿐만 아니라 LDAP 서버에 있는 그룹에도 적용됩니다.

사전 요구 사항

- 동기화 구성 파일을 생성합니다.

절차

- LDAP 그룹의 하위 집합을 OpenShift Container Platform과 동기화하려면 다음 명령을 사용하십시오.

```
$ oc adm groups sync --whitelist=<whitelist_file> \
                     --sync-config=config.yaml      \
                     --confirm
```

```
$ oc adm groups sync --blacklist=<blacklist_file> \
                     --sync-config=config.yaml      \
                     --confirm
```

```
$ oc adm groups sync <group_unique_identifier>    \
                     --sync-config=config.yaml      \
                     --confirm
```

```
$ oc adm groups sync <group_unique_identifier>  \
                     --whitelist=<whitelist_file> \
                     --blacklist=<blacklist_file> \
                     --sync-config=config.yaml    \
                     --confirm
```

```
$ oc adm groups sync --type=openshift           \
                     --whitelist=<whitelist_file> \
                     --sync-config=config.yaml    \
                     --confirm
```

참고

기본적으로 모든 그룹 동기화 작업은 시험 실행이므로 OpenShift Container Platform 그룹 레코드를 변경하려면 **oc adm groups sync** 명령에 **--confirm** 플래그를 설정해야 합니다.

17.3. 그룹 정리 작업 실행

그룹을 생성한 LDAP 서버의 레코드가 더 이상 존재하지 않는 경우, 관리자는 OpenShift Container Platform 레코드에서 그룹을 제거하도록 선택할 수도 있습니다. 정리 작업에는 동기화 작업에 사용한 것과 동일한 동기화 구성 파일과 허용 목록 또는 차단 목록을 사용할 수 있습니다.

예를 들면 다음과 같습니다.

-

```
$ oc adm prune groups --sync-config=/path/to/ldap-sync-config.yaml --confirm
```

```
$ oc adm prune groups --whitelist=/path/to/whitelist.txt --sync-config=/path/to/ldap-sync-config.yaml --confirm
```

```
$ oc adm prune groups --blacklist=/path/to/blacklist.txt --sync-config=/path/to/ldap-sync-config.yaml --confirm
```

17.4. LDAP 그룹 자동 동기화

cron 작업을 구성하여 정기적으로 LDAP 그룹을 동기화할 수 있습니다.

사전 요구 사항

- **cluster-admin** 역할의 사용자로 클러스터에 액세스할 수 있어야 합니다.
- LDAP ID 공급자(IDP)를 구성했습니다.
  이 절차에서는 **ldap-secret**이라는 LDAP 시크릿과 **ca-config-map** 이라는 구성 맵을 생성했다고 가정합니다.

절차

1. cron 작업이 실행될 프로젝트를 생성합니다.

   ```
   $ oc new-project ldap-sync 1
   ```

   1 이 절차에서는 **ldap-sync**라는 프로젝트를 사용합니다.

2. LDAP ID 공급자를 구성할 때 생성한 시크릿 및 구성 맵을 찾아 이 새 프로젝트에 복사합니다.
   시크릿 및 구성 맵은 **openshift-config** 프로젝트에 있으며 새 **ldap-sync** 프로젝트에 복사해야 합니다.

3. 서비스 계정을 정의합니다.

   예: ldap-sync-service-account.yaml

   ```
   kind: ServiceAccount
   apiVersion: v1
   metadata:
     name: ldap-group-syncer
     namespace: ldap-sync
   ```

4. 서비스 계정을 생성합니다.

   ```
   $ oc create -f ldap-sync-service-account.yaml
   ```

5. 클러스터 역할을 정의합니다.

   예: ldap-sync-cluster-role.yaml

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   ```

```
kind: ClusterRole
metadata:
  name: ldap-group-syncer
rules:
  - apiGroups:
      - ''
      - user.openshift.io
    resources:
      - groups
    verbs:
      - get
      - list
      - create
      - update
```

6. 클러스터 역할을 생성합니다.

```
$ oc create -f ldap-sync-cluster-role.yaml
```

7. 클러스터 역할을 서비스 계정에 바인딩할 클러스터 역할 바인딩을 정의합니다.

예: ldap-sync-cluster-role-binding.yaml

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ldap-group-syncer
subjects:
  - kind: ServiceAccount
    name: ldap-group-syncer          1
    namespace: ldap-sync
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ldap-group-syncer            2
```

1. 이 절차의 앞부분에서 생성한 서비스 계정에 대한 참조입니다.
2. 이 절차의 앞부분에서 생성한 클러스터 역할에 대한 참조입니다.

8. 클러스터 역할 바인딩을 생성합니다.

```
$ oc create -f ldap-sync-cluster-role-binding.yaml
```

9. 동기화 구성 파일을 지정하는 구성 맵을 정의합니다.

예: ldap-sync-config-map.yaml

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
```

```
data:
  sync.yaml: |                                  1
    kind: LDAPSyncConfig
    apiVersion: v1
    url: ldaps://10.0.0.0:389                   2
    insecure: false
    bindDN: cn=admin,dc=example,dc=com          3
    bindPassword:
      file: "/etc/secrets/bindPassword"
    ca: /etc/ldap-ca/ca.crt
    rfc2307:                                    4
      groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"   5
        scope: sub
        filter: "(objectClass=groupOfMembers)"
        derefAliases: never
        pageSize: 0
      groupUIDAttribute: dn
      groupNameAttributes: [ cn ]
      groupMembershipAttributes: [ member ]
      usersQuery:
        baseDN: "ou=users,dc=example,dc=com"    6
        scope: sub
        derefAliases: never
        pageSize: 0
      userUIDAttribute: dn
      userNameAttributes: [ uid ]
      tolerateMemberNotFoundErrors: false
      tolerateMemberOutOfScopeErrors: false
```

1. 동기화 구성 파일을 정의합니다.
2. URL을 지정합니다.
3. **bindDN**을 지정합니다.
4. 이 예에서는 RFC2307 스키마를 사용하고 필요에 따라 값을 조정합니다. 다른 스키마를 사용할 수도 있습니다.
5. **groupsQuery**에 대한 **baseDN**을 지정합니다.
6. **usersQuery**에 대한 **baseDN**을 지정합니다.

10. config map을 생성합니다.

```
$ oc create -f ldap-sync-config-map.yaml
```

11. cron 작업을 정의합니다.

예: ldap-sync-cron-job.yaml

```
kind: CronJob
apiVersion: batch/v1
metadata:
```

```
  name: ldap-group-syncer
  namespace: ldap-sync
spec:                                                    (1)
  schedule: "*/30 * * * *"                               (2)
  concurrencyPolicy: Forbid
  jobTemplate:
    spec:
      backoffLimit: 0
      ttlSecondsAfterFinished: 1800                      (3)
      template:
        spec:
          containers:
            - name: ldap-group-sync
              image: "registry.redhat.io/openshift4/ose-cli:latest"
              command:
                - "/bin/bash"
                - "-c"
                - "oc adm groups sync --sync-config=/etc/config/sync.yaml --confirm"   (4)
              volumeMounts:
                - mountPath: "/etc/config"
                  name: "ldap-sync-volume"
                - mountPath: "/etc/secrets"
                  name: "ldap-bind-password"
                - mountPath: "/etc/ldap-ca"
                  name: "ldap-ca"
          volumes:
            - name: "ldap-sync-volume"
              configMap:
                name: "ldap-group-syncer"
            - name: "ldap-bind-password"
              secret:
                secretName: "ldap-secret"                (5)
            - name: "ldap-ca"
              configMap:
                name: "ca-config-map"                    (6)
          restartPolicy: "Never"
          terminationGracePeriodSeconds: 30
          activeDeadlineSeconds: 500
          dnsPolicy: "ClusterFirst"
          serviceAccountName: "ldap-group-syncer"
```

(1) cron 작업의 설정을 구성합니다. cron 작업 설정에 대한 자세한 내용은 "Cron 작업 생성"에서 참조하십시오.

(2) cron 형식으로 지정된 작업의 스케줄입니다. 이 예제 cron 작업은 30분마다 실행됩니다. 필요에 따라 빈도를 조정하여 동기화를 실행하는 데 걸리는 시간을 고려합니다.

(3) 완료된 작업을 유지하는 시간(초)입니다. 이전 실패한 작업을 정리하고 불필요한 경고를 방지하기 위해 작업 일정의 기간과 일치해야 합니다. 자세한 내용은 Kubernetes 문서의 TTL-after-finished Controller 에서 참조하십시오.

(4) 실행할 cron 작업의 LDAP 동기화 명령입니다. 구성 맵에 정의된 동기화 구성 파일에서 전달합니다.

(5) 이 시크릿은 LDAP IDP를 구성할 때 생성되었습니다.

6 이 구성 맵은 LDAP IDP 구성 시 생성되었습니다.

12. cron 작업을 생성합니다.

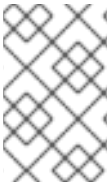
```
$ oc create -f ldap-sync-cron-job.yaml
```

추가 리소스

- LDAP ID 공급자 구성
- cron 작업 생성

17.5. LDAP 그룹 동기화의 예

이 섹션에는 RFC 2307, Active Directory, 보강된 Active Directory 스키마에 대한 예가 포함되어 있습니다.



참고

이러한 예에서는 모든 사용자를 해당 그룹의 직접 멤버로 가정합니다. 특히, 어떠한 그룹도 다른 그룹의 멤버가 될 수 없습니다. 중첩 그룹을 동기화하는 방법에 대한 정보는 중첩 멤버십 동기화 예를 참조하십시오.

17.5.1. RFC 2307 스키마를 사용한 그룹 동기화

RFC 2307 스키마의 경우 다음 예제에서는 두 개의 멤버가 있는 **admins** 그룹을 동기화합니다. **Jane** 와 **Jim**. 예제에서는 다음을 설명합니다.

- 그룹 및 사용자를 LDAP 서버에 추가하는 방법
- 동기화 후 OpenShift Container Platform에 생성되는 결과 그룹 레코드

참고

이러한 예에서는 모든 사용자를 해당 그룹의 직접 멤버로 가정합니다. 특히, 어떠한 그룹도 다른 그룹의 멤버가 될 수 없습니다. 중첩 그룹을 동기화하는 방법에 대한 정보는 중첩 멤버십 동기화 예를 참조하십시오.

RFC 2307 스키마에서는 사용자(Jane 및 Jim)와 그룹 모두 LDAP 서버에 최상위 항목으로 존재하며, 그룹 멤버십은 그룹 속성에 저장됩니다. 다음의 **ldif** 조각에서는 이 스키마의 사용자 및 그룹을 정의합니다.

RFC 2307 스키마를 사용하는 LDAP 항목: rfc2307.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users
dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Jane
sn: Smith
displayName: Jane Smith
```

```
mail: jane.smith@example.com
dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups
dn: cn=admins,ou=groups,dc=example,dc=com (1)
objectClass: groupOfNames
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com (2)
member: cn=Jim,ou=users,dc=example,dc=com
```

1. 그룹은 LDAP 서버의 최상위 항목입니다.
2. 그룹 멤버와 함께 그룹 속성이 식별을 위한 참조 정보로 나열됩니다.

사전 요구 사항

- 구성 파일을 생성합니다.

절차

- **rfc2307_config.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync --sync-config=rfc2307_config.yaml --confirm
```

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

rfc2307_config.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400 (1)
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com (2)
    openshift.io/ldap.url: LDAP_SERVER_IP:389 (3)
  creationTimestamp:
  name: admins (4)
users: (5)
- jane.smith@example.com
- jim.adams@example.com
```

1. 이 OpenShift Container Platform 그룹이 LDAP 서버와 마지막으로 동기화된 시간으로, ISO 6801 형식으로 되어 있습니다.

2 LDAP 서버에서 그룹의 고유 식별자입니다.

3 이 그룹의 레코드가 저장된 LDAP 서버의 IP 주소 및 호스트입니다.

4 동기화 파일에서 지정한 그룹의 이름입니다.

5 동기화 파일에서 지정한 대로 이름이 지정된 그룹 멤버에 해당하는 사용자입니다.

17.5.2. RFC2307 스키마와 사용자 정의 이름 매핑을 사용한 그룹 동기화

사용자 정의 이름 매핑과 그룹을 동기화하면 구성 파일이 이러한 매핑을 포함하도록 아래와 같이 변경됩니다.

사용자 정의 이름 매핑과 함께 RFC 2307 스키마를 사용하는 LDAP 동기화 구성: rfc2307_config_user_defined.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
groupUIDNameMapping:
  "cn=admins,ou=groups,dc=example,dc=com": Administrators 1
rfc2307:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
        pageSize: 0
    groupUIDAttribute: dn 2
    groupNameAttributes: [ cn ] 3
    groupMembershipAttributes: [ member ]
    usersQuery:
        baseDN: "ou=users,dc=example,dc=com"
        scope: sub
        derefAliases: never
        pageSize: 0
    userUIDAttribute: dn 4
    userNameAttributes: [ mail ]
    tolerateMemberNotFoundErrors: false
    tolerateMemberOutOfScopeErrors: false
```

1 사용자 정의 이름 매핑입니다.

2 사용자 정의 이름 매핑에서 키에 사용되는 고유 식별자 속성입니다. groupUIDAttribute로 DN을 사용할 때는 **groupsQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.

3 사용자 정의 이름 매핑에 고유 식별자가 없는 경우 OpenShift Container Platform 그룹의 이름을 지정하는 속성입니다.

4 LDAP 서버에서 사용자를 고유하게 식별하는 속성입니다. userUIDAttribute로 DN을 사용할 때는 **usersQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.

사전 요구 사항

- 구성 파일을 생성합니다.

절차

- **rfc2307_config_user_defined.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync --sync-config=rfc2307_config_user_defined.yaml --confirm
```

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

rfc2307_config_user_defined.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com
    openshift.io/ldap.url: LDAP_SERVER_IP:389
  creationTimestamp:
  name: Administrators 1
users:
- jane.smith@example.com
- jim.adams@example.com
```

1 사용자 정의 이름 매핑에 의해 지정된 그룹의 이름입니다.

17.5.3. RFC 2307과 사용자 정의 오류 허용 오차를 사용한 그룹 동기화

기본적으로 동기화 중인 그룹에 멤버 쿼리에 정의된 범위를 벗어나는 항목이 있는 멤버가 포함된 경우, 그룹 동기화에 오류가 발생하여 실패합니다.

```
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with dn="<user-dn>" would search outside of the base dn specified (dn="<base-dn>")".
```

이러한 오류는 대부분 **usersQuery** 필드의 **baseDN**이 잘못 구성되었음을 나타냅니다. 그러나 **baseDN**에에 의도적으로 일부 그룹 멤버가 포함되어 있지 않은 경우, **tolerateMemberOutOfScopeErrors:true**를 설정하면 그룹을 계속 동기화할 수 있습니다. 범위를 벗어난 멤버는 무시됩니다.

마찬가지로 그룹 동기화 프로세스에서 그룹 멤버를 찾지 못하면 오류와 함께 실패합니다.

```
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with base dn="<user-dn>" refers to a non-existent entry".
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with base dn="<user-dn>" and filter "<filter>" did not return any results".
```

이러한 오류는 대부분 **usersQuery** 필드가 잘못 구성되었음을 나타냅니다. 그러나 그룹에 누락된 것으로 알려진 멤버 항목이 포함된 경우, **tolerateMemberNotFoundErrors:true**로 설정하면 그룹을 계속 동기화할 수 있습니다. 문제가 있는 멤버는 무시됩니다.

주의

LDAP 그룹 동기화에 오류 허용을 사용하면 동기화 프로세스에서 문제가 있는 멤버 항목을 무시합니다. LDAP 그룹 동기화가 올바르게 구성되어 있지 않으면 동기화된 OpenShift Container Platform 그룹에서 멤버가 누락될 수 있습니다.

그룹 멤버십에 문제가 있는 RFC 2307 스키마를 사용하는 LDAP 항목: rfc2307_problematic_users.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users
dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups
dn: cn=admins,ou=groups,dc=example,dc=com
objectClass: groupOfNames
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com
member: cn=Jim,ou=users,dc=example,dc=com
member: cn=INVALID,ou=users,dc=example,dc=com (1)
member: cn=Jim,ou=OUTOFSCOPE,dc=example,dc=com (2)
```

1. LDAP 서버에 존재하지 않는 멤버입니다.
2. 동기화 작업에 대한 사용자 쿼리에 있을 수 있지만 **baseDN**에 없는 멤버입니다.

위 예제의 오류를 허용하려면 동기화 구성 파일에 다음을 추가해야 합니다.

오류를 허용하는 RFC 2307 스키마를 사용하는 LDAP 동기화 구성: rfc2307_config_tolerating.yaml

```
-
```

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
rfc2307:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
    groupUIDAttribute: dn
    groupNameAttributes: [ cn ]
    groupMembershipAttributes: [ member ]
    usersQuery:
        baseDN: "ou=users,dc=example,dc=com"
        scope: sub
        derefAliases: never
    userUIDAttribute: dn (1)
    userNameAttributes: [ mail ]
    tolerateMemberNotFoundErrors: true (2)
    tolerateMemberOutOfScopeErrors: true (3)
```

1. LDAP 서버에서 사용자를 고유하게 식별하는 속성입니다. userUIDAttribute로 DN을 사용할 때는 **usersQuery** 필터를 지정할 수 없습니다. 세분화된 필터링의 경우 허용 목록/차단 목록 메서드를 사용하십시오.

2. **true**인 경우 동기화 작업에서는 일부 멤버가 없는 그룹을 허용하고 LDAP 항목이 없는 멤버를 무시합니다. 그룹 멤버를 찾을 수 없는 경우 동기화 작업의 기본 동작은 실패하는 것입니다.

3. **true**인 경우 동기화 작업에서는 일부 멤버가 **usersQuery** 기본 DN에 지정된 사용자 범위를 벗어나는 그룹을 허용하고, 멤버 조회 범위를 벗어나는 멤버를 무시합니다. 그룹 멤버가 범위를 벗어나는 경우 동기화 작업의 기본 동작은 실패하는 것입니다.

사전 요구 사항

- 구성 파일을 생성합니다.

절차

- **rfc2307_config_tolerating.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync --sync-config=rfc2307_config_tolerating.yaml --confirm
```

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

rfc2307_config.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com
    openshift.io/ldap.url: LDAP_SERVER_IP:389
  creationTimestamp:
```

```
  name: admins
users: 1
- jane.smith@example.com
- jim.adams@example.com
```

1 동기화 파일에서 지정한 대로 그룹 멤버에 해당하는 사용자입니다. 조회에 허용되는 오류가 발생한 멤버가 없습니다.

17.5.4. Active Directory 스키마를 사용하여 그룹 동기화

Active Directory 스키마에서는 두 사용자(Jane 및 Jim) 모두 LDAP 서버에 최상위 항목으로 존재하며, 그룹 멤버십은 사용자 속성에 저장됩니다. 다음의 **ldif** 조각에서는 이 스키마의 사용자 및 그룹을 정의합니다.

Active Directory 스키마를 사용하는 LDAP 항목: active_directory.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
memberOf: admins 1

dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
memberOf: admins
```

1 사용자의 그룹 멤버십은 사용자의 속성으로 나열되며, 그룹은 서버에 항목으로 존재하지 않습니다. **memberOf** 속성이 사용자의 리터럴 속성이 아닐 수도 있습니다. 일부 LDAP 서버에서는 검색 중 생성되어 클라이언트에 반환되지만 데이터베이스에는 커밋되지 않습니다.

사전 요구 사항

- 구성 파일을 생성합니다.

절차

- **active_directory_config.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync --sync-config=active_directory_config.yaml --confirm
```

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

active_directory_config.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400 (1)
    openshift.io/ldap.uid: admins (2)
    openshift.io/ldap.url: LDAP_SERVER_IP:389 (3)
  creationTimestamp:
  name: admins (4)
users: (5)
- jane.smith@example.com
- jim.adams@example.com
```

(1) 이 OpenShift Container Platform 그룹이 LDAP 서버와 마지막으로 동기화된 시간으로, ISO 6801 형식으로 되어 있습니다.

(2) LDAP 서버에서 그룹의 고유 식별자입니다.

(3) 이 그룹의 레코드가 저장된 LDAP 서버의 IP 주소 및 호스트입니다.

(4) LDAP 서버에 나열된 그룹의 이름입니다.

(5) 동기화 파일에서 지정한 대로 이름이 지정된 그룹 멤버에 해당하는 사용자입니다.

17.5.5. 보강된 Active Directory 스키마를 사용하여 그룹 동기화

보강된 Active Directory 스키마에서는 사용자(Jane 및 Jim)와 그룹 모두 LDAP 서버에 최상위 항목으로 존재하며, 그룹 멤버십은 사용자 속성에 저장됩니다. 다음의 **ldif** 조각에서는 이 스키마의 사용자 및 그룹을 정의합니다.

보강된 Active Directory 스키마를 사용하는 LDAP 항목: augmented_active_directory.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
```

```
mail: jane.smith@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com (1)

dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

dn: cn=admins,ou=groups,dc=example,dc=com (2)
objectClass: groupOfNames
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com
member: cn=Jim,ou=users,dc=example,dc=com
```

(1) 사용자의 그룹 멤버십이 사용자 속성으로 나열됩니다.

(2) 그룹은 LDAP 서버의 최상위 항목입니다.

사전 요구 사항

- 구성 파일을 생성합니다.

절차

- **augmented_active_directory_config.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync --sync-config=augmented_active_directory_config.yaml --confirm
```

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

augmented_active_directory_config.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400 (1)
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com (2)
    openshift.io/ldap.url: LDAP_SERVER_IP:389 (3)
  creationTimestamp:
```

```
  name: admins 4
users: 5
- jane.smith@example.com
- jim.adams@example.com
```

1. 이 OpenShift Container Platform 그룹이 LDAP 서버와 마지막으로 동기화된 시간으로, ISO 6801 형식으로 되어 있습니다.
2. LDAP 서버에서 그룹의 고유 식별자입니다.
3. 이 그룹의 레코드가 저장된 LDAP 서버의 IP 주소 및 호스트입니다.
4. 동기화 파일에서 지정한 그룹의 이름입니다.
5. 동기화 파일에서 지정한 대로 이름이 지정된 그룹 멤버에 해당하는 사용자입니다.

17.5.5.1. LDAP 중첩 멤버십 동기화의 예

OpenShift Container Platform의 그룹은 중첩되지 않습니다. 데이터를 사용하려면 LDAP 서버에서 그룹 멤버십을 평면화해야 합니다. Microsoft의 Active Directory 서버에서는 OID **1.2.840.113556.1.4.1941**이 있는 **LDAP_MATCHING_RULE_IN_CHAIN** 규칙을 통해 이 기능을 지원합니다. 또한 이 일치 규칙을 사용하는 경우에는 명시적으로 허용된 그룹만 동기화할 수 있습니다.

이 섹션에는 한 명의 사용자 **Jane**과 하나의 그룹 **otheradmins**가 멤버인 **admins**라는 그룹을 동기화하는 보강된 Active Directory 스키마에 대한 예가 있습니다. **otheradmins** 그룹에는 하나의 사용자 멤버가 있습니다. **Jim**. 이 예제에서는 다음을 설명합니다.

- 그룹 및 사용자를 LDAP 서버에 추가하는 방법
- LDAP 동기화 구성 파일의 내용
- 동기화 후 OpenShift Container Platform에 생성되는 결과 그룹 레코드

보강된 Active Directory 스키마에서는 사용자(**Jane** 및 **Jim**)와 그룹 모두 LDAP 서버에 최상위 항목으로 존재하며, 그룹 멤버십은 사용자 또는 그룹 속성에 저장됩니다. 다음의 **ldif** 조각에서는 이 스키마의 사용자 및 그룹을 정의합니다.

보강된 Active Directory 스키마를 중첩 멤버와 함께 사용하는 LDAP 항목: augmented_active_directory_nested.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com 1
```

```
dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
memberOf: cn=otheradmins,ou=groups,dc=example,dc=com 2

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

dn: cn=admins,ou=groups,dc=example,dc=com 3
objectClass: group
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com
member: cn=otheradmins,ou=groups,dc=example,dc=com

dn: cn=otheradmins,ou=groups,dc=example,dc=com 4
objectClass: group
cn: otheradmins
owner: cn=admin,dc=example,dc=com
description: Other System Administrators
memberOf: cn=admins,ou=groups,dc=example,dc=com 5 6
member: cn=Jim,ou=users,dc=example,dc=com
```

1 2 5 사용자 및 그룹의 멤버십은 오브젝트 속성으로 나열됩니다.

3 4 그룹은 LDAP 서버의 최상위 항목입니다.

6 **otheradmins** 그룹은 **admins** 그룹의 멤버입니다.

중첩 그룹을 Active Directory와 동기화하는 경우 사용자 항목 및 그룹 항목에 대한 LDAP 쿼리 정의와 내부 OpenShift Container Platform 그룹 레코드에서 해당 항목을 표시하는 속성을 제공해야 합니다. 또한 이 구성에는 특정 변경이 필요합니다.

- **oc adm groups sync** 명령에서 그룹을 명시적으로 허용해야 합니다.
- **LDAP_MATCHING_RULE_IN_CHAIN** 규칙을 준수하려면 사용자의 **groupMembershipAttributes**에 **"memberOf:1.2.840.113556.1.4.1941:"**이 포함되어야 합니다.
- **groupUIDAttribute**를 **dn**으로 설정해야 합니다.
- **groupsQuery**의 경우
  - **filter**를 설정하지 않아야 합니다.
  - 유효한 **derefAliases**를 설정해야 합니다.

- 해당 값이 무시되므로 **baseDN**을 설정하지 않아야 합니다.
- 해당 값이 무시되므로 **scope**를 설정하지 않아야 합니다.

명확히 하기 위해, OpenShift Container Platform에서 생성한 그룹의 경우 사용자용 필드 또는 관리자용 필드에 가급적 고유 이름 이외의 속성을 사용해야 합니다. 예를 들면 이메일로 OpenShift Container Platform 그룹의 사용자를 구분하고, 그룹 이름을 공통 이름으로 사용합니다. 다음 구성 파일에서는 이러한 관계를 생성합니다.

중첩 멤버와 함께 보강된 Active Directory 스키마를 사용하는 LDAP 동기화 구성: augmented_active_directory_config_nested.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
augmentedActiveDirectory:
  groupsQuery: (1)
    derefAliases: never
    pageSize: 0
  groupUIDAttribute: dn (2)
  groupNameAttributes: [ cn ] (3)
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=person)
    pageSize: 0
  userNameAttributes: [ mail ] (4)
  groupMembershipAttributes: [ "memberOf:1.2.840.113556.1.4.1941:" ] (5)
```

(1) **groupsQuery** 필터를 지정할 수 없습니다. **groupsQuery** 기본 DN 및 범위 값이 무시됩니다. **groupsQuery** 에서 유효한 **derefAliases** 를 설정해야 합니다.

(2) LDAP 서버에서 그룹을 고유하게 식별하는 속성입니다. **dn**으로 설정해야 합니다.

(3) 그룹 이름으로 사용할 속성입니다.

(4) OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다. **mail** 또는 **sAMAccountName** 은 대부분의 설치에서 선호됩니다.

(5) 멤버십 정보를 저장하는 사용자 속성입니다. **LDAP_MATCHING_RULE_IN_CHAIN** 사용에 유의하십시오.

사전 요구 사항

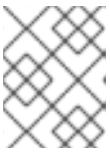
- 구성 파일을 생성합니다.

절차

- **augmented_active_directory_config_nested.yaml** 파일과의 동기화를 실행합니다.

```
$ oc adm groups sync \
  'cn=admins,ou=groups,dc=example,dc=com' \
  --sync-config=augmented_active_directory_config_nested.yaml \
```

```
--confirm
```



참고

cn=admins,ou=groups,dc=example,dc=com 그룹을 명시적으로 허용해야 합니다.

OpenShift Container Platform은 위 동기화 작업의 결과로 다음과 같은 그룹 레코드를 만듭니다.

augmented_active_directory_config_nested.yaml 파일을 사용하여 생성된 OpenShift Container Platform 그룹

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  annotations:
    openshift.io/ldap.sync-time: 2015-10-13T10:08:38-0400 (1)
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com (2)
    openshift.io/ldap.url: LDAP_SERVER_IP:389 (3)
  creationTimestamp:
  name: admins (4)
users: (5)
- jane.smith@example.com
- jim.adams@example.com
```

(1) 이 OpenShift Container Platform 그룹이 LDAP 서버와 마지막으로 동기화된 시간으로, ISO 6801 형식으로 되어 있습니다.

(2) LDAP 서버에서 그룹의 고유 식별자입니다.

(3) 이 그룹의 레코드가 저장된 LDAP 서버의 IP 주소 및 호스트입니다.

(4) 동기화 파일에서 지정한 그룹의 이름입니다.

(5) 동기화 파일에서 지정한 대로 이름이 지정된 그룹 멤버에 해당하는 사용자입니다. 그룹 멤버십이 Microsoft Active Directory Server에 의해 평면화되었으므로 중첩 그룹의 멤버가 포함됩니다.

17.6. LDAP 동기화 구성 사양

구성 파일의 오브젝트 사양은 다음과 같습니다. 필드는 스키마 오브젝트에 따라 달라집니다. 예를 들어, v1.ActiveDirectoryConfig에는 **groupsQuery** 필드가 없지만 v1.RFC2307Config 및 v1.AugmentedActiveDirectoryConfig에는 있습니다.

중요

바이너리 속성은 지원되지 않습니다. LDAP 서버에서 제공하는 모든 속성 데이터는 UTF-8 인코딩 문자열 형식이어야 합니다. 예를 들면 **objectGUID**와 같은 바이너리 속성을 ID 속성으로 사용하지 마십시오. 그 대신 **sAMAccountName** 또는 **userPrincipalName**과 같은 문자열 속성을 사용해야 합니다.

17.6.1. v1.LDAPSyncConfig

LDAPSyncConfig에는 LDAP 그룹 동기화를 정의하는 데 필요한 구성 옵션이 있습니다.

이름	설명	스키마
kind	이 오브젝트가 나타내는 REST 리소스에 해당하는 문자열 값입니다. 서버는 클라이언트에서 요청을 제출한 끝점에서 이를 유추할 수 있습니다. CamelCase로 업데이트할 수 없습니다. 자세한 내용은 https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#types-kinds를 참조하십시오.	문자열
apiVersion	버전이 지정된 이 오브젝트 표현의 스키마를 정의합니다. 서버는 인식된 스키마를 최신 내부 값으로 변환해야 하며, 인식되지 않는 값을 거부할 수 있습니다. 자세한 내용은 https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#resources를 참조하십시오.	문자열
url	호스트는 연결할 LDAP 서버의 스키마, 호스트 및 포트입니다 (**scheme://host:port**).	문자열
bindDN	LDAP 서버에 바인딩할 선택적 DN입니다.	문자열
bindPassword	검색 단계에서 바인딩할 선택적 암호입니다.	v1.StringSource
insecure	**true**인 경우 연결에서 TLS를 사용하지 않아야 함을 나타냅니다. **false**인 경우 **ldaps://** URL은 TLS를 사용하여 연결되고, **ldap://** URL은 https://tools.ietf.org/html/rfc2830에 지정된 StartTLS를 사용하여 TLS 연결로 업그레이드됩니다. **insecure**를 **true**로 설정하는 경우 **ldaps://** URL 스키마를 사용할 수 없습니다.	부울

이름	설명	스키마
ca	서버에 요청할 때 사용하는 신뢰할 수 있는 인증 기관 번들 옵션입니다. 비어있는 경우 기본 시스템 루트가 사용됩니다.	문자열
groupUIDNameMapping	LDAP 그룹 UID를 OpenShift Container Platform 그룹 이름에 직접 매핑하는 옵션입니다.	오브젝트
rfc2307	RFC2307과 유사한 방식으로 설정된 LDAP 서버에서 데이터를 추출하는 구성이 있습니다. 최상위 그룹 및 사용자 항목이 있고 멤버를 나열하는 그룹 항목 다중값 속성에 따라 그룹 멤버십이 결정됩니다.	v1.RFC2307Config
activeDirectory	Active Directory 방식과 유사한 방식으로 설정된 LDAP 서버에서 데이터를 추출하는 구성이 있습니다. 최상위 사용자 항목이 있고 멤버가 속한 그룹을 나열하는 멤버 다중값 속성에 따라 그룹 멤버십이 결정됩니다.	v1.ActiveDirectoryConfig
augmentedActiveDirectory	위에서 설명한 Active Directory 방식과 유사한 방식으로 설정된 LDAP 서버에서 데이터를 추출하는 구성이 있습니다. 단, 최상위 그룹 항목이 존재하고 여기에 그룹 멤버십이 아닌 메타데이터를 보관합니다.	v1.AugmentedActiveDirectoryConfig

17.6.2. v1.StringSource

StringSource를 사용하면 문자열을 인라인으로 지정하거나 환경 변수 또는 파일을 통해 외부에서 지정할 수 있습니다. 문자열 값만 포함하는 경우 간단한 JSON 문자열로 마샬링됩니다.

이름	설명	스키마
value	일반 텍스트 값 또는 **keyFile**이 지정된 경우 암호화된 값을 지정합니다.	문자열
env	일반 텍스트 값을 포함하는 환경 변수 또는 **keyFile**이 지정된 경우 암호화된 값을 지정합니다.	문자열

이름	설명	스키마
file	일반 텍스트 값이 포함된 파일 또는 **keyFile**이 지정된 경우 암호화된 값을 참조합니다.	문자열
keyFile	값을 해독하는 데 사용할 키가 들어 있는 파일을 참조합니다.	문자열

17.6.3. v1.LDAPQuery

LDAPQuery에는 LDAP 쿼리를 빌드하는 데 필요한 옵션이 있습니다.

이름	설명	스키마
baseDN	모든 검색을 시작하는 디렉터리 분기의 DN입니다.	문자열
scope	검색 범위 옵션입니다. **base**(기본 오브젝트만), **one**(기본 수준의 모든 오브젝트), **sub**(전체 하위 트리)를 사용할 수 있습니다. 설정하지 않는 경우 기본값은 **sub**입니다.	문자열
derefAliases	별칭과 관련된 검색 동작 옵션입니다. **never**(별칭을 역참조하지 않음), **search**(검색에서만 역참조), **base**(기본 오브젝트를 찾을 때만 역참조), **always**(항상 역참조)가 될 수 있습니다. 설정하지 않는 경우 기본값은 **always**입니다.	문자열
timeout	응답 대기를 종료하기 전에 서버에 대한 요청을 처리 중 상태로 유지할 수 있는 시간(초)이 있습니다. 이 값이 **0**이면 클라이언트 쪽 제한이 적용되지 않습니다.	정수
filter	기본 DN이 있는 LDAP 서버에서 관련 항목을 모두 검색하는 유효한 LDAP 검색 필터입니다.	문자열
pageSize	LDAP 항목으로 측정한 최대 기본 페이지 크기입니다. 페이지 크기가 **0**이면 페이징이 수행되지 않습니다.	정수

17.6.4. v1.RFC2307Config

RFC2307Config에는 RFC2307 스키마를 사용하여 LDAP 그룹 동기화에서 LDAP 서버와 상호 작용하는 방식을 정의하는 데 필요한 구성 옵션이 있습니다.

이름	설명	스키마
groupsQuery	그룹 항목을 반환하는 LDAP 쿼리용 템플릿이 있습니다.	v1.LDAPQuery
groupUIDAttribute	고유 식별자로 해석되는 LDAP 그룹 항목의 속성을 정의합니다 (**ldapGroupUID**).	문자열
groupNameAttributes	OpenShift Container Platform 그룹에 사용할 이름으로 해석되는 LDAP 그룹 항목의 속성을 정의합니다.	문자열 배열
groupMembershipAttributes	멤버로 해석되는 LDAP 그룹 항목의 속성을 정의합니다. 해당 속성에 포함된 값을 **UserUIDAttribute**로 쿼리할 수 있어야 합니다.	문자열 배열
usersQuery	사용자 항목을 반환하는 LDAP 쿼리용 템플릿이 있습니다.	v1.LDAPQuery
userUIDAttribute	고유 식별자로 해석되는 LDAP 사용자 항목의 속성을 정의합니다. **GroupMembershipAttributes**에 있는 값과 일치해야 합니다.	문자열
userNameAttributes	OpenShift Container Platform 사용자 이름으로 사용할 LDAP 사용자 항목의 속성을 순서대로 정의합니다. 값이 비어 있지 않은 첫 번째 속성이 사용됩니다. 이 값은 **LDAPPasswordIdentityProvider**의 **PreferredUsername** 설정과 일치해야 합니다. OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다. **mail** 또는 **sAMAccountName** 은 대부분의 설치에서 선호됩니다.	문자열 배열

이름	설명	스키마
tolerateMemberNotFoundErrors	누락된 사용자 항목이 있는 경우 LDAP 동기화 작업의 동작을 결정합니다. **true**인 경우 찾지 못한 사용자에 대한 LDAP 쿼리가 허용되며, 오류만 기록됩니다. **false**인 경우 사용자에 대한 쿼리에서 아무것도 찾지 못하면 LDAP 동기화 작업이 실패합니다. 기본값은 **false**입니다. 이 플래그를 **true**로 설정하여 LDAP 동기화 작업을 잘못 구성하면 그룹 멤버십이 제거될 수 있으므로 이 플래그를 주의해서 사용하십시오.	부울
tolerateMemberOutOfScopeErrors	범위를 벗어난 사용자 항목이 있는 경우 LDAP 동기화 작업의 동작을 결정합니다. **true**인 경우 모든 사용자 쿼리에 지정된 기본 DN을 벗어나는 사용자의 LDAP 쿼리가 허용되며 오류만 기록됩니다. **false**인 경우 사용자 쿼리가 모든 사용자 쿼리에서 지정하는 기본 DN 외부에서 검색하면 LDAP 동기화 작업이 실패합니다. 이 플래그를 **true**로 설정하여 LDAP 동기화 작업을 잘못 구성하면 그룹에서 사용자가 누락될 수 있으므로 이 플래그를 주의해서 사용하는 것이 좋습니다.	부울

17.6.5. v1.ActiveDirectoryConfig

ActiveDirectoryConfig에는 Active Directory 스키마를 사용하여 LDAP 그룹 동기화에서 LDAP 서버와 상호 작용하는 방식을 정의하는 데 필요한 구성 옵션이 있습니다.

이름	설명	스키마
usersQuery	사용자 항목을 반환하는 LDAP 쿼리용 템플릿이 있습니다.	v1.LDAPQuery
userNameAttributes	OpenShift Container Platform 사용자 이름으로 해석되는 LDAP 사용자 항목의 속성을 정의합니다. OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다. **mail** 또는 **sAMAccountName** 은 대부분의 설치에서 선호됩니다.	문자열 배열

이름	설명	스키마
groupMembershipAttributes	멤버가 속한 그룹으로 해석되는 LDAP 사용자 항목의 속성을 정의합니다.	문자열 배열

17.6.6. v1.AugmentedActiveDirectoryConfig

AugmentedActiveDirectoryConfig에는 보강된 Active Directory 스키마를 사용하여 LDAP 그룹 동기화에서 LDAP 서버와 상호 작용하는 방식을 정의하는 데 필요한 구성 옵션이 있습니다.

이름	설명	스키마
usersQuery	사용자 항목을 반환하는 LDAP 쿼리용 템플릿이 있습니다.	v1.LDAPQuery
userNameAttributes	OpenShift Container Platform 사용자 이름으로 해석되는 LDAP 사용자 항목의 속성을 정의합니다. OpenShift Container Platform 그룹 레코드에서 사용자 이름으로 사용할 속성입니다. **mail** 또는 **sAMAccountName** 은 대부분의 설치에서 선호됩니다.	문자열 배열
groupMembershipAttributes	멤버가 속한 그룹으로 해석되는 LDAP 사용자 항목의 속성을 정의합니다.	문자열 배열
groupsQuery	그룹 항목을 반환하는 LDAP 쿼리용 템플릿이 있습니다.	v1.LDAPQuery
groupUIDAttribute	고유 식별자로 해석되는 LDAP 그룹 항목의 속성을 정의합니다 (**IdapGroupUID**).	문자열
groupNameAttributes	OpenShift Container Platform 그룹에 사용할 이름으로 해석되는 LDAP 그룹 항목의 속성을 정의합니다.	문자열 배열

18장. 클라우드 공급자 인증 정보 관리

18.1. CLOUD CREDENTIAL OPERATOR 정보

CCO(Cloud Credential Operator)는 클라우드 공급자 인증 정보를 CRD(사용자 지정 리소스 정의)로 관리합니다. CCO가 **CredentialsRequest** 사용자 정의 리소스(CR)에 동기화되어 OpenShift Container Platform 구성 요소가 클러스터를 실행하는 데 필요한 특정 권한으로 클라우드 공급자 인증 정보를 요청할 수 있습니다.

install-config.yaml 파일에서 **credentialsMode** 매개변수에 다양한 값을 설정하면 CCO를 여러 모드에서 작동하도록 구성할 수 있습니다. 모드가 지정되지 않거나 **credentialsMode** 매개변수가 빈 문자열(**""**)로 설정되면 CCO는 기본 모드에서 작동합니다.

18.1.1. 모드

install-config.yaml 파일에서 **credentialsMode** 매개변수의 다른 값을 설정하면 *mint*, *passthrough* 또는 *manual* 모드에서 CCO를 작동하도록 구성할 수 있습니다. 이러한 옵션은 CCO에서 클라우드 인증 정보를 사용하여 클러스터의 **CredentialsRequest** CR을 처리하는 방법에 투명성과 유연성을 제공하고 조직의 보안 요구 사항에 맞게 CCO를 구성할 수 있습니다. 모든 클라우드 공급자에서 모든 CCO 모드가 지원되는 것은 아닙니다.

- Mint: Mint 모드에서 CCO는 제공된 관리자 수준 클라우드 인증 정보를 사용하여 필요한 특정 권한만으로 클러스터의 구성 요소에 대한 새 인증 정보를 생성합니다.

  참고

  Mint 모드는 CCO가 사용할 기본 및 권장 모범 사례 설정입니다.

- 패스스루: passthrough 모드에서 CCO는 제공된 클라우드 인증 정보를 클라우드 인증 정보를 요청하는 구성 요소에 전달합니다.

- 수동: 수동 모드에서 사용자는 CCO 대신 클라우드 인증 정보를 관리합니다.

  - AWS STS가 포함된 수동: 수동 모드에서는 AWS STS(Amazon Web Services Secure Token Service)를 사용하도록 AWS 클러스터를 구성할 수 있습니다. 이 구성을 통해 CCO는 다른 구성 요소에 임시 인증 정보를 사용합니다.

표 18.1. CCO 모드 지원 매트릭스

클라우드 공급자	Mint	Passthrough	Manual
AWS(Amazon Web Services)	X	X	X
Microsoft Azure		X	X
GCP(Google Cloud Platform)	X	X	X
Red Hat OpenStack Platform (RHOSP)		X	
RHV(Red Hat Virtualization)		X	

클라우드 공급자	Mint	Passthrough	Manual
VMware vSphere		X	

18.1.2. 기본 동작

여러 모드가 지원되는 플랫폼의 경우(AWS, Azure 및 GCP) CCO가 기본 모드에서 실행되면 **CredentialsRequest** CR을 충분히 처리할 수 있는 모드를 결정하도록 제공된 인증 정보를 동적으로 확인합니다.

기본적으로 CCO는 인증 정보가 기본 작동 모드인 mint 모드에 충분한지 결정하고 해당 인증 정보를 사용하여 클러스터의 구성 요소에 적절한 인증 정보를 생성합니다. 인증 정보가 Mint 모드에 충분하지 않으면 passthrough 모드에 충분한지 여부를 결정합니다. 인증 정보가 passthrough 모드에 충분하지 않으면 CCO에서 **CredentialsRequest** CR을 적절하게 처리할 수 없습니다.

설치 중에 제공된 인증 정보가 충분하지 않다고 판단되면 설치에 실패합니다. AWS의 경우 설치 프로그램은 프로세스 초기에 실패하고 필요한 권한이 누락된 것을 나타냅니다. 다른 공급자는 오류가 발생할 때까지 오류 발생 원인에 대한 구체적인 정보를 제공하지 않을 수 있습니다.

성공적인 설치 후 인증 정보가 변경되고 CCO가 새 인증 정보가 충분하지 않다고 판단하면 CCO에서 새 **CredentialsRequest** CR에 조건을 배치하여 충분하지 않은 인증 정보로 인해 이를 처리할 수 없음을 나타냅니다.

충분하지 않은 인증 정보 문제를 해결하려면 충분한 권한이 있는 인증 정보를 제공하십시오. 설치 중에 오류가 발생한 경우 다시 설치해보십시오. 새로운 **CredentialsRequest** CR 문제의 경우 CCO가 CR을 다시 처리할 때까지 기다립니다. 다른 방법으로 AWS,Azure 및 GCP에 대해 IAM을 수동으로 생성할 수 있습니다.

18.1.3. 추가 리소스

- Cloud Credential Operator의 Cluster Operators 참조 페이지

18.2. MINT 모드 사용

Mint 모드는 AWS(Amazon Web Services) 및 GCP(Google Cloud Platform)에서 지원됩니다.

Mint 모드는 지원되는 플랫폼의 기본 모드입니다. 이 모드에서 CCO(Cloud Credential Operator)는 제공된 관리자 수준 클라우드 인증 정보를 사용하여 필요한 특정 권한만 사용하여 클러스터의 구성 요소에 대한 새 인증 정보를 생성합니다.

설치 후 인증 정보를 제거하지 않으면 CCO에서 저장하고 사용하여 클러스터의 구성 요소에 대한 **CredentialsRequest** CR을 처리하고 필요한 특정 권한으로 각각에 대한 새 인증 정보를 생성합니다. mint 모드에서 클라우드 인증 정보의 지속적인 조정을 통해 업그레이드 등을 진행하기 위해 추가 인증 정보 또는 권한이 필요한 작업을 수행할 수 있습니다.

Mint 모드는 클러스터 **kube-system** 네임스페이스에 관리자 수준 인증 정보를 저장합니다. 이 접근 방식이 조직의 보안 요구 사항을 충족하지 않는 경우 AWS 또는 GCP 의 *kube-system 프로젝트에 관리자 수준 시크릿을 저장하기 위한 대체* 를 참조하십시오.

18.2.1. Mint 모드 권한 요구사항

Mint 모드에서 CCO를 사용하는 경우 제공한 인증 정보가 OpenShift Container Platform을 실행 중이거나 설치하는 클라우드의 요구 사항을 충족하는지 확인하십시오. 제공된 인증 정보가 mint 모드에 충분하지 않으면 CCO에서 IAM 사용자를 생성할 수 없습니다.

18.2.1.1. AWS(Amazon Web Services) 권한

AWS에서 Mint 모드에 제공하는 인증 정보에는 다음과 같은 권한이 있어야 합니다.

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:SimulatePrincipalPolicy**

18.2.1.2. GCP(Google Cloud Platform) 권한

GCP에서 Mint 모드에 제공하는 인증 정보에는 다음과 같은 권한이 있어야 합니다.

- **resourcemanager.projects.get**
- **serviceusage.services.list**
- **iam.serviceAccountKeys.create**
- **iam.serviceAccountKeys.delete**
- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.roles.get**
- **resourcemanager.projects.getIamPolicy**
- **resourcemanager.projects.setIamPolicy**

18.2.2. 관리자 인증 정보 루트 시크릿 형식

각 클라우드 공급자는 규칙에 따라 **kube-system** 네임 스페이스의 인증 정보 루트 시크릿을 사용하며, 이

를 통해 모든 인증 정보 요청을 충족하고 해당 암호를 만드는 데 사용됩니다. 이 작업은 *mint 모드*를 사용하여 새 인증 정보를 생성하거나 *패스스루 모드*를 사용하여 인증 정보 루트 시크릿을 복사하여 수행됩니다.

시크릿의 형식은 클라우드에 따라 다르며 각 **CredentialsRequest** 시크릿에도 사용됩니다.

Amazon Web Services (AWS) 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <base64-encoded_access_key_id>
  aws_secret_access_key: <base64-encoded_secret_access_key>
```

Google Cloud Platform (GCP) 시크릿 형식

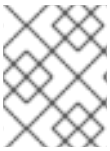
```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

18.2.3. 관리자 수준 인증 정보를 제거하거나 교체하는 Mint 모드

현재 이 모드는 AWS 및 GCP에서만 지원됩니다.

이 모드에서 사용자는 일반 Mint 모드와 마찬가지로 관리자 수준 인증 정보를 사용하여 OpenShift Container Platform을 설치합니다. 그러나 이 프로세스에서는 설치 후 클러스터에서 관리자 수준 인증 정보 시크릿을 제거합니다.

관리자는 모든 **CredentialsRequest** 오브젝트에 필요한 사용 권한이 있고 따라서 내용을 변경해야 하는 경우가 아니면 관리자 수준의 자격 증명이 필요 없음을 확인하기 위해 Cloud Credential Operator가 읽기 전용 자격 증명을 스스로 요청하도록 할 수 있습니다. 연결된 인증 정보를 제거한 후 원하는 경우 기본 클라우드에서 삭제하거나 비활성화할 수 있습니다.



참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

관리자 수준 인증 정보는 클러스터에 영구적으로 저장되지 않습니다.

이러한 단계를 따르려면 짧은 기간 동안 클러스터의 관리자 수준 인증 정보가 필요합니다. 또한 각 업그레이드에 대해 관리자 수준 인증 정보로 시크릿을 수동으로 복원해야 합니다.

18.2.3.1. 클라우드 공급자 인증 정보를 수동으로 교체

어떠한 이유로 클라우드 공급자 인증 정보가 변경되면 CCO(Cloud Credential Operator)에서 클라우드 공급자 인증 정보를 관리하기 위해 사용하는 시크릿을 수동으로 업데이트해야 합니다.

클라우드 인증 정보를 교체하는 프로세스는 CCO가 사용하도록 구성된 모드에 따라 달라집니다. Mint 모드를 사용하는 클러스터의 인증 정보를 교체한 후 삭제된 인증 정보를 통해 생성된 구성 요소 인증 정보를 수동으로 제거해야 합니다.

사전 요구 사항

- 클러스터는 다음을 사용하는 CCO 모드로 클라우드 인증 정보 교체를 수동으로 지원하는 플랫폼에 설치됩니다.
  - Mint 모드의 경우 AWS(Amazon Web Services) 및 GCP(Google Cloud Platform)가 지원됩니다.
- 클라우드 공급자와 인터페이스에 사용되는 인증 정보를 변경했습니다.
- 새 인증 정보에는 클러스터에서 사용할 수 있도록 구성된 모드 CCO에 대한 충분한 권한이 있습니다.

절차

1. 웹 콘솔의 **Administrator** 모드에서 **Workloads → Secrets**로 이동합니다.
2. **Secrets** 페이지의 표에서 클라우드 공급자의 루트 시크릿을 찾습니다.

플랫폼	시크릿 이름
AWS	**aws-creds**
GCP	**gcp-credentials**

3. 시크릿과 동일한 행에서 **옵션** 메뉴  를 클릭하고 **시크릿 편집**을 선택합니다.
4. **Value** 필드의 내용을 기록합니다. 이 정보를 사용하여 인증서를 업데이트한 후 값이 다른지 확인할 수 있습니다.
5. 클라우드 공급자에 대한 새로운 인증 정보를 사용하여 **Value** 필드의 텍스트를 업데이트한 다음 **저장**을 클릭합니다.
6. 개별 **CredentialsRequest** 개체에서 참조하는 각 구성 요소 시크릿을 삭제합니다.
   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.
   b. 참조되는 모든 구성 요소 시크릿의 이름과 네임스페이스를 가져옵니다.

```
$ oc -n openshift-cloud-credential-operator get CredentialsRequest \
  -o json | jq -r '.items[] | select (.spec.providerSpec.kind=="<provider_spec>") |
.spec.secretRef'
```

   여기서 **<provider_spec** >은 클라우드 공급자의 해당 값입니다.

   - AWS: **AWSProviderSpec**
   - GCP: **GCPProviderSpec**

AWS의 부분 예제 출력

```
{
  "name": "ebs-cloud-credentials",
  "namespace": "openshift-cluster-csi-drivers"
}
{
  "name": "cloud-credential-operator-iam-ro-creds",
  "namespace": "openshift-cloud-credential-operator"
}
```

c. 참조된 각 구성 요소 시크릿을 삭제합니다.

```
$ oc delete secret <secret_name> \ 1
  -n <secret_namespace> 2
```

1 보안 이름을 지정합니다.

2 보안이 포함된 네임스페이스를 지정합니다.

AWS 시크릿 삭제 예

```
$ oc delete secret ebs-cloud-credentials -n openshift-cluster-csi-drivers
```

공급자 콘솔에서 인증 정보를 수동으로 삭제할 필요가 없습니다. 참조된 구성 요소 시크릿을 삭제하면 CCO가 플랫폼에서 기존 인증 정보를 삭제하고 새 인증서를 생성합니다.

검증

인증 정보가 변경되었는지 확인하려면 다음을 수행하십시오.

1. 웹 콘솔의 **Administrator** 모드에서 **Workloads → Secrets**로 이동합니다.
2. **Value** 필드의 내용이 변경되었는지 확인합니다.

18.2.3.2. 클라우드 공급자 인증 정보 제거

Mint 모드에서 CCO(Cloud Credential Operator)를 사용하여 OpenShift Container Platform 클러스터를 설치한 후 클러스터의 **kube-system** 네임스페이스에서 관리자 수준 인증 정보 시크릿을 제거할 수 있습니다. 관리자 수준 인증 정보는 업그레이드와 같은 승격된 권한이 필요한 변경 시에만 필요합니다.

참고

z-stream 외 업그레이드 이전에는 관리자 수준 인증 정보를 사용하여 인증 정보 시크릿을 복원해야 합니다. 인증 정보가 없으면 업그레이드가 차단될 수 있습니다.

사전 요구 사항

- 클러스터는 CCO에서 클라우드 인증 정보 제거를 지원하는 플랫폼에 설치되어 있습니다. 지원되는 플랫폼은 AWS 및 GCP입니다.

절차

1. 웹 콘솔의 **Administrator** 모드에서 **Workloads → Secrets**로 이동합니다.
2. **Secrets** 페이지의 표에서 클라우드 공급자의 루트 시크릿을 찾습니다.

플랫폼	시크릿 이름
AWS	**aws-creds**
GCP	**gcp-credentials**

3. 시크릿과 동일한 행에서 **옵션** 메뉴  를 클릭하고 **시크릿 삭제**를 선택합니다.

18.2.4. 추가 리소스

- AWS의 kube-system 프로젝트에서 관리자 수준 시크릿을 저장하는 대안
- GCP의 kube-system 프로젝트에서 관리자 수준 시크릿을 저장하는 대안

18.3. PASSTHROUGH 모드 사용

Pssthrough 모드는 AWS(Amazon Web Services), Microsoft Azure, GCP(Google Cloud Platform), RHOSP(Red Hat OpenStack Platform), RHV(Red Hat Virtualization) 및 VMware vSphere에서 지원됩니다.

Passthrough 모드에서 CCO(Cloud Credential Operator)는 제공된 클라우드 인증 정보를 클라우드 인증 정보를 요청하는 구성 요소에 전달합니다. 인증 정보에는 설치를 수행하고 클러스터의 구성 요소에 필요한 작업을 완료할 수 있는 권한이 있어야 하지만 새 인증 정보를 생성할 필요가 없습니다. CCO는 passthrough 모드로 추가 제한된 범위 인증 정보를 생성하지 않습니다.

18.3.1. Passthrough 모드 권한 요구사항

Passthrough 모드에서 CCO를 사용하는 경우 제공한 인증 정보가 OpenShift Container Platform을 실행 중이거나 설치하는 클라우드의 요구 사항을 충족하는지 확인하십시오. 제공된 인증 정보에서 CCO가 **CredentialsRequest** CR을 생성하는 구성 요소에 충분하지 않으면 구성 요소가 권한이 없는 API를 호출하려고 할 때 오류를 보고합니다.

18.3.1.1. AWS(Amazon Web Services) 권한

AWS에서 passthrough 모드로 제공하는 인증 정보에는 실행 또는 설치 중인 OpenShift Container Platform 버전에 필요한 모든 **CredentialsRequest** CR에 대한 요청된 권한이 있어야 합니다.

필요한 **CredentialsRequest** CR을 찾으려면 AWS용 IAM을 수동으로 생성을 참조하십시오.

18.3.1.2. Microsoft Azure 권한

Azure에서 passthrough 모드에 제공하는 인증 정보에는 실행 중 또는 설치 중인 OpenShift Container Platform 버전에 필요한 모든 **CredentialsRequest** CR에 대한 요청된 권한이 있어야 합니다.

필요한 **CredentialsRequest** CR을 찾으려면 Azure용 IAM을 수동으로 생성을 참조하십시오.

18.3.1.3. GCP(Google Cloud Platform) 권한

GCP에서 passthrough 모드로 제공하는 인증 정보에는 실행 중 또는 설치 중인 OpenShift Container Platform 버전에 필요한 모든 **CredentialsRequest** CR에 대한 요청된 권한이 있어야 합니다.

필요한 **CredentialsRequest** CR을 찾으려면 GCP용 IAM을 수동으로 생성을 참조하십시오.

18.3.1.4. RHOSP(Red Hat OpenStack Platform) 권한

RHOSP에 OpenShift Container Platform 클러스터를 설치하려면 CCO에 **member** 사용자 역할의 권한이 있는 인증 정보가 필요합니다.

18.3.1.5. RHV(Red Hat Virtualization) 권한

RHV에 OpenShift Container Platform 클러스터를 설치하려면 CCO에 다음 권한이 있는 인증 정보가 필요합니다.

- **DiskOperator**
- **DiskCreator**
- **UserTemplateBasedVm**
- **TemplateOwner**
- **TemplateCreator**
- OpenShift Container Platform 배포를 대상으로 하는 특정 클러스터의 **ClusterAdmin**

18.3.1.6. VMware vSphere 권한

VMware vSphere에 OpenShift Container Platform 클러스터를 설치하려면 CCO에 다음 vSphere 권한이 있는 인증 정보가 필요합니다.

표 18.2. 필수 vSphere 권한

카테고리	권한
데이터 저장소	*공간 할당*
폴더	*폴더 생성, 폴더 삭제*
vSphere 태그 지정	모든 권한
네트워크	*네트워크 할당*
리소스 이름	*리소스 풀에 가상 머신 할당*
프로필 중심 스토리지	모든 권한
vApp	모든 권한
가상 머신	모든 권한

18.3.2. 관리자 인증 정보 루트 시크릿 형식

각 클라우드 공급자는 규칙에 따라 **kube-system** 네임 스페이스의 인증 정보 루트 시크릿을 사용하며, 이를 통해 모든 인증 정보 요청을 충족하고 해당 암호를 만드는 데 사용됩니다. 이 작업은 *mint* 모드를 사용하여 새 인증 정보를 생성하거나 *패스스루* 모드를 사용하여 인증 정보 루트 시크릿을 복사하여 수행됩니다.

시크릿의 형식은 클라우드에 따라 다르며 각 **CredentialsRequest** 시크릿에도 사용됩니다.

Amazon Web Services (AWS) 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <base64-encoded_access_key_id>
  aws_secret_access_key: <base64-encoded_secret_access_key>
```

Microsoft Azure 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: azure-credentials
stringData:
  azure_subscription_id: <base64-encoded_subscription_id>
  azure_client_id: <base64-encoded_client_id>
  azure_client_secret: <base64-encoded_client_secret>
  azure_tenant_id: <base64-encoded_tenant_id>
  azure_resource_prefix: <base64-encoded_resource_prefix>
  azure_resourcegroup: <base64-encoded_resource_group>
  azure_region: <base64-encoded_region>
```

Microsoft Azure에서 인증 정보 시크릿 형식에는 각 클러스터 설치에 대해 임의로 생성된 클러스터의 인프라 ID를 포함해야 하는 두 가지 속성이 포함되어 있습니다. 이 값은 매니페스트 생성을 실행한 후에 찾을 수 있습니다.

```
$ cat .openshift_install_state.json | jq '."*installconfig.ClusterID".InfraID' -r
```

출력 예

```
mycluster-2mpcn
```

이 값은 다음과 같이 시크릿 데이터에 사용됩니다.

```
azure_resource_prefix: mycluster-2mpcn
azure_resourcegroup: mycluster-2mpcn-rg
```

Google Cloud Platform (GCP) 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

RHOSP(Red Hat OpenStack Platform) 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: openstack-credentials
data:
  clouds.yaml: <base64-encoded_cloud_creds>
  clouds.conf: <base64-encoded_cloud_creds_init>
```

RHV(Red Hat Virtualization) 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: ovirt-credentials
data:
  ovirt_url: <base64-encoded_url>
  ovirt_username: <base64-encoded_username>
  ovirt_password: <base64-encoded_password>
  ovirt_insecure: <base64-encoded_insecure>
  ovirt_ca_bundle: <base64-encoded_ca_bundle>
```

VMware vSphere 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: vsphere-creds
data:
 vsphere.openshift.example.com.username: <base64-encoded_username>
 vsphere.openshift.example.com.password: <base64-encoded_password>
```

18.3.3. Passthrough 모드 인증 정보 유지 관리

클러스터가 업그레이드되어 **CredentialsRequest** CR이 시간이 지남에 따라 변경되는 경우 요구사항을 충족하도록 passthrough 모드 인증 정보를 수동으로 업데이트해야 합니다. 업그레이드 중 인증 정보 문제가 발생하지 않도록 하려면 업그레이드 전에 새 OpenShift Container Platform 버전에 대해 릴리스 이미지의 **CredentialsRequest** CR을 확인하십시오. 클라우드 공급자에 필요한 **CredentialsRequest** CR을 찾으려면 AWS,Azure 또는 GCP에 대해 *IAM 수동 생성*을 참조하십시오.

18.3.3.1. 클라우드 공급자 인증 정보를 수동으로 교체

어떠한 이유로 클라우드 공급자 인증 정보가 변경되면 CCO(Cloud Credential Operator)에서 클라우드 공급자 인증 정보를 관리하기 위해 사용하는 시크릿을 수동으로 업데이트해야 합니다.

클라우드 인증 정보를 교체하는 프로세스는 CCO가 사용하도록 구성된 모드에 따라 달라집니다. Mint 모드를 사용하는 클러스터의 인증 정보를 교체한 후 삭제된 인증 정보를 통해 생성된 구성 요소 인증 정보를 수동으로 제거해야 합니다.

사전 요구 사항

- 클러스터는 다음을 사용하는 CCO 모드로 클라우드 인증 정보 교체를 수동으로 지원하는 플랫폼에 설치됩니다.
  - Passthrough 모드의 경우 AWS(Amazon Web Services), Microsoft Azure, GCP(Google Cloud Platform), RHOSP(Red Hat OpenStack Platform), RHV(Red Hat Virtualization) 및 VMware vSphere가 지원됩니다.
- 클라우드 공급자와 인터페이스에 사용되는 인증 정보를 변경했습니다.
- 새 인증 정보에는 클러스터에서 사용할 수 있도록 구성된 모드 CCO에 대한 충분한 권한이 있습니다.

절차

1. 웹 콘솔의 **Administrator** 모드에서 **Workloads → Secrets**로 이동합니다.
2. **Secrets** 페이지의 표에서 클라우드 공급자의 루트 시크릿을 찾습니다.

플랫폼	시크릿 이름
AWS	**aws-creds**
Azure	**azure-credentials**
GCP	**gcp-credentials**
RHOSP	**openstack-credentials**
RHV	**ovirt-credentials**
VMware vSphere	**vsphere-creds**

3. 시크릿과 동일한 행에서 **옵션** 메뉴  를 클릭하고 **시크릿 편집**을 선택합니다.
4. **Value** 필드의 내용을 기록합니다. 이 정보를 사용하여 인증서를 업데이트한 후 값이 다른지 확인할 수 있습니다.
5. 클라우드 공급자에 대한 새로운 인증 정보를 사용하여 **Value** 필드의 텍스트를 업데이트한 다음 **저장**을 클릭합니다.
6. vSphere CSI Driver Operator가 활성화되어 있지 않은 vSphere 클러스터의 인증 정보를 업데이트하는 경우 Kubernetes 컨트롤러 관리자의 롤아웃을 강제 적용하여 업데이트된 인증 정보를 적용해야 합니다.

참고

vSphere CSI Driver Operator가 활성화된 경우 이 단계가 필요하지 않습니다.

업데이트된 vSphere 인증 정보를 적용하려면 OpenShift Container Platform CLI에 **cluster-admin** 역할의 사용자로 로그인하고 다음 명령을 실행합니다.

```
$ oc patch kubecontrollermanager cluster \
  -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date )"'"}}' \
  --type=merge
```

인증 정보가 출시되는 동안 Kubernetes Controller Manager Operator의 상태는 **Progressing=true** 로 보고합니다. 상태를 보려면 다음 명령을 실행합니다.

```
$ oc get co kube-controller-manager
```

검증

인증 정보가 변경되었는지 확인하려면 다음을 수행하십시오.

1. 웹 콘솔의 **Administrator** 모드에서 **Workloads → Secrets**로 이동합니다.
2. **Value** 필드의 내용이 변경되었는지 확인합니다.

추가 리소스

- vSphere CSI Driver Operator

18.3.4. 설치 후 권한 감소

passthrough 모드를 사용할 때 각 구성 요소에는 다른 모든 구성 요소가 사용하는 것과 동일한 권한이 있습니다. 설치 후 권한을 줄이지 않으면 모든 구성 요소에 설치 프로그램 실행에 필요한 광범위한 권한이 있습니다.

설치 후 사용 중인 OpenShift Container Platform 버전에 대한 릴리스 이미지의 **CredentialsRequest** CR에 정의된 대로 인증 정보의 권한을 클러스터를 실행하는 데 필요한 권한으로만 줄일 수 있습니다.

AWS, Azure 또는 GCP에 필요한 **CredentialsRequest** CR을 찾고 CCO가 사용하는 권한을 변경하는 방법을 알아보려면 AWS, Azure 또는 GCP의 *IAM 수동 생성*을 참조하십시오.

18.3.5. 추가 리소스

- AWS의 IAM 수동 생성
- 수동으로 Azure 용 IAM 생성
- 수동으로 GCP 용 IAM 생성

18.4. 수동 모드 사용

수동 모드는 AWS(Amazon Web Services), Microsoft Azure 및 GCP(Google Cloud Platform)에서 지원됩니다.

수동 모드에서 사용자는 CCO(Cloud Credential Operator) 대신 클라우드 인증 정보를 관리합니다. 이 모

드를 사용하려면 실행 또는 설치 중인 OpenShift Container Platform 버전의 릴리스 이미지에서 **CredentialsRequest** CR을 검사하고, 기본 클라우드 공급자에 해당 자격 증명을 생성한 다음, 클러스터 클라우드 공급자에 대한 모든 **CredentialsRequest** CR을 충족하기 위해 올바른 네임스페이스에 Kubernetes 보안을 생성해야 합니다.

수동 모드를 사용하면 각 클러스터 구성 요소에는 클러스터에 관리자 수준 인증 정보를 저장하지 않고 필요한 권한만 보유할 수 있습니다. 이 모드에서도 AWS 공용 IAM 끝점에 연결할 필요가 없습니다. 그러나 업그레이드할 때마다 새 릴리스 이미지로 권한을 수동으로 조정해야 합니다.

수동 모드를 사용하도록 클라우드 공급자를 구성하는 방법에 대한 자세한 내용은 AWS, Azure 또는 GCP의 *IAM 수동 생성*을 참조하십시오.

18.4.1. AWS STS를 사용하는 수동 모드

AWS STS(Amazon Web Services Secure Token Service) 를 사용하도록 수동 모드에서 AWS 클러스터를 구성할 수 있습니다. 이 구성을 통해 CCO는 다른 구성 요소에 임시 인증 정보를 사용합니다.

18.4.2. 수동으로 유지 관리되는 인증 정보로 클러스터 업그레이드

CCO(Cloud Credential Operator) 수동으로 유지 관리되는 인증 정보가 있는 클러스터의 **Upgradable** 상태는 기본적으로 **False** 입니다.

- 마이너 릴리스(예: 4.7에서 4.8로)의 경우 이 상태는 업데이트된 권한을 처리하고 **CloudCredential** 리소스에 주석을 달 때까지 업그레이드하여 다음 버전에 대한 필요에 따라 권한이 업데이트됨을 나타냅니다. 이 주석은 **Upgradable** 상태를 **True**로 변경합니다.
- 예를 들어 4.8.9에서 4.8.10으로 z-stream 릴리스의 경우 권한이 추가되거나 변경되지 않으므로 업그레이드가 차단되지 않습니다.

수동으로 유지 관리되는 인증 정보로 클러스터를 업그레이드하기 전에 업그레이드할 릴리스 이미지에 대한 새 인증 정보를 생성해야 합니다. 또한 기존 인증 정보에 필요한 권한을 검토하고 해당 구성 요소에 대한 새 릴리스에 새 권한 요구 사항을 수용해야 합니다.

절차

1. 새 릴리스에 대한 **CredentialsRequest** 사용자 지정 리소스를 추출하고 검사합니다.
   클라우드 공급자용 설치 콘텐츠의 "수동으로 IAM 생성" 섹션에서는 클라우드에 필요한 인증 정보를 획득하고 사용하는 방법을 설명합니다.
2. 클러스터에서 수동으로 유지 관리되는 인증 정보를 업데이트합니다.
   - 새 릴리스 이미지에서 추가한 **CredentialsRequest** 사용자 정의 리소스에 대한 새 시크릿을 생성합니다.
   - 시크릿에 저장된 기존 인증 정보에 대한 **CredentialsRequest** 사용자 정의 리소스가 권한 요구 사항이 변경된 경우 필요에 따라 권한을 업데이트합니다.
3. 모든 보안이 새 릴리스에 대해 올바른 경우 클러스터를 업그레이드할 준비가 되었음을 나타냅니다.
   a. **cluster-admin** 역할의 사용자로 OpenShift Container Platform CLI에 로그인합니다.
   b. **CloudCredential** 리소스를 편집하여 **metadata** 필드 내에 **upgradeable-to** 주석을 추가합니다.

```
$ oc edit cloudcredential cluster
```

추가할 텍스트

```
...
  metadata:
    annotations:
      cloudcredential.openshift.io/upgradeable-to: <version_number>
...
```

여기서 **<version_number>**는 **x.y.z** 형식으로 업그레이드할 버전입니다. 예를 들어 OpenShift Container Platform 4.8.2 의 경우 **4.8.2**입니다.

주석을 추가한 후 업그레이드 가능 상태가 변경되는 데 몇 분이 소요될 수 있습니다.

4. CCO를 업그레이드할 수 있는지 확인합니다.
   a. 웹 콘솔의 **관리자** 화면에서 **관리자 → 클러스터 설정**으로 이동합니다.
   b. CCO 상태 세부 정보를 보려면 **Cluster Operators** 목록에서 **cloud-credential**을 클릭합니다.
   c. **Conditions** 섹션의 **Upgradeable** 상태가 **False**인 경우 **upgradeable-to 주석**에 오타 오류가 없는지 확인합니다.

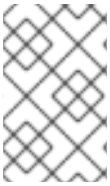
Conditions 섹션의 **Upgradeable** 상태가 **True** 이면 OpenShift Container Platform 업그레이드를 시작할 수 있습니다.

18.4.3. 추가 리소스

- AWS의 IAM 수동 생성
- 수동으로 Azure 용 IAM 생성
- 수동으로 GCP 용 IAM 생성
- AWS STS를 사용하는 수동 모드 사용

18.5. AMAZON WEB SERVICES SECURE TOKEN SERVICE에서 수동 모드 사용

STS를 사용하는 수동 모드는 AWS(Amazon Web Services)에서 지원됩니다.



참고

이 인증 정보 전략은 새 OpenShift Container Platform 클러스터에 대해서만 지원되며 설치 중에 구성해야 합니다. 이 기능을 사용하기 위해 다른 인증 정보 전략을 사용하는 기존 클러스터를 재구성할 수 없습니다.

18.5.1. AWS Secure Token Service를 사용한 수동 모드 정보

STS를 사용하는 수동 모드에서 개별 OpenShift Container Platform 클러스터 구성 요소는 AWS STS(Secure Token Service)를 사용하여 단기적으로 제한된 권한 보안 인증 정보를 제공하는 구성 요소 IAM 역할을 할당합니다. 이러한 인증 정보는 AWS API 호출을 수행하는 각 구성 요소에 특정적인 IAM 역할과 연결됩니다.

새로운 인증 정보 및 새로고침된 인증 정보에 대한 요청은 AWS IAM 역할과 함께 적절하게 구성된 AWS

IAM OIDC(OpenID Connect) ID 공급자를 사용하여 자동화됩니다. OpenShift Container Platform은 AWS IAM에서 신뢰하는 서비스 계정 토큰에 서명하고 Pod에 프로젝션하고 인증에 사용할 수 있습니다. 토큰은 1시간 후에 새로 고쳐집니다.

그림 18.1. STS 인증 흐름

STS를 사용하여 수동 모드를 사용하면 개별 OpenShift Container Platform 구성 요소에 제공되는 AWS 인증 정보의 내용이 변경됩니다.

수명이 긴 인증 정보를 사용하는 AWS 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <target-namespace> 1
  name: <target-secret-name> 2
data:
  aws_access_key_id: <base64-encoded-access-key-id>
  aws_secret_access_key: <base64-encoded-secret-access-key>
```

1. 구성 요소의 네임스페이스입니다.
2. 구성 요소 시크릿의 이름입니다.

STS를 사용하는 AWS 시크릿 형식

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <target-namespace> 1
  name: <target-secret-name> 2
stringData:
  credentials: |-
    [default]
    role_name: <operator-role-name> 3
    web_identity_token_file: <path-to-token> 4
```

1. 구성 요소의 네임스페이스입니다.
2. 구성 요소 시크릿의 이름입니다.
3. 구성 요소의 IAM 역할입니다.

4 Pod 내의 서비스 계정 토큰 경로입니다. 관례상 이는 OpenShift Container Platform 구성 요소의 **/var/run/secrets/openshift/serviceaccount/token** 입니다.

18.5.2. STS를 사용하여 수동 모드에 구성된 OpenShift Container Platform 클러스터 설치

STS를 사용하여 수동 모드에서 CCO(Cloud Credential Operator)를 사용하도록 구성된 클러스터를 설치하려면 다음을 수행합니다.

1. Cloud Credential Operator 유틸리티를 구성합니다.
2. 필요한 AWS 리소스를 개별적으로 또는 단일 명령을 사용하여 생성합니다.
3. OpenShift Container Platform 설치 프로그램을 실행합니다.
4. 클러스터가 수명이 짧은 인증 정보를 사용하고 있는지 확인합니다.

참고

STS를 사용할 때 클러스터가 수동 모드에서 작동하므로 필요한 권한으로 구성 요소에 대한 새 인증 정보를 생성할 수 없습니다. OpenShift Container Platform의 다른 마이너 버전으로 업그레이드할 때 종종 새로운 AWS 권한 요구 사항이 있습니다. STS를 사용하는 클러스터를 업그레이드하기 전에 클러스터 관리자는 AWS 권한이 기존 구성 요소에 충분하고 새 구성 요소에서 사용할 수 있는지 수동으로 확인해야 합니다.

18.5.2.1. Cloud Credential Operator 유틸리티 구성

STS를 사용하여 CCO(Cloud Credential Operator)가 수동 모드에서 작동하는 경우 클러스터 외부에서 클라우드 인증 정보를 생성하고 관리하려면 CCO 유틸리티(**ccoctl**) 바이너리를 추출하고 준비합니다.

참고

ccoctl은 Linux 환경에서 실행해야 하는 Linux 바이너리입니다.

절차

1. OpenShift Container Platform 릴리스 이미지를 가져옵니다.

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. OpenShift Container Platform 릴리스 이미지에서 CCO 컨테이너 이미지를 가져옵니다.

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE)
```

참고

$RELEASE_IMAGE 의 아키텍처가 **ccoctl** 툴을 사용할 환경의 아키텍처와 일치하는지 확인합니다.

3. OpenShift Container Platform 릴리스 이미지 내의 CCO 컨테이너 이미지에서 **ccoctl** 바이너리를 추출합니다.

```
$ oc image extract $CCO_IMAGE --file="/usr/bin/ccoctl" -a ~/.pull-secret
```

4. **ccoctl**을 실행 가능하게 하려면 권한을 변경합니다.

```
$ chmod 775 ccoctl
```

검증

- **ccoctl**을 사용할 준비가 되었는지 확인하려면 도움말 파일을 표시합니다.

```
$ ccoctl aws --help
```

ccoctl aws --help의 출력 :

```
Creating/updating/deleting cloud credentials objects for AWS cloud


Usage:
  ccoctl aws [command]


Available Commands:
  create-all              Create all the required credentials objects
  create-iam-roles        Create IAM roles
  create-identity-provider Create IAM identity provider
  create-key-pair         Create a key pair
  delete                  Delete credentials objects

Flags:
  -h, --help   help for aws

Use "ccoctl aws [command] --help" for more information about a command.
```

18.5.2.2. Cloud Credential Operator 유틸리티를 사용하여 AWS 리소스 생성

CCO 유틸리티(**ccoctl**)를 사용하여 필요한 AWS 리소스를 개별적으로 생성하거나 단일 명령으로 생성할 수 있습니다.

18.5.2.2.1. 개별적으로 AWS 리소스 생성

AWS 리소스를 수정하기 전에 **ccoctl** 툴에서 생성하는 JSON 파일을 검토해야 하거나 **ccoctl** 툴에서 AWS 리소스를 생성하는 프로세스에서 조직의 요구 사항을 자동으로 충족하지 않는 경우 AWS 리소스를 개별적으로 생성할 수 있습니다. 예를 들어 이 옵션은 다양한 사용자 또는 부서 간에 이러한 리소스를 만드는 책임을 공유하는 조직에 유용할 수 있습니다.

그렇지 않으면 **ccoctl aws create-all** 명령을 사용하여 AWS 리소스를 자동으로 생성할 수 있습니다.

참고

기본적으로 **ccoctl**은 명령이 실행되는 디렉터리에 오브젝트를 생성합니다. 다른 디렉터리에 오브젝트를 생성하려면 **--output-dir** 플래그를 사용합니다. 이 절차에서는 & **lt;path_to_ccoctl_output_dir** >을 사용하여 이 디렉터리를 참조합니다.

일부 **ccoctl** 명령은 AWS API를 호출하여 AWS 리소스를 생성하거나 수정합니다. **--dry-run** 플래그를 사용하여 API 호출을 방지할 수 있습니다. 이 플래그를 사용하면 로컬 파일 시스템에 JSON 파일이 생성됩니다. JSON 파일을 검토 및 수정한 다음 **--cli-input-json** 매개변수를 사용하여 AWS CLI 툴로 적용할 수 있습니다.

사전 요구 사항

- **ccoctl** 바이너리를 추출하고 준비합니다.

절차

1. 클러스터의 OpenID Connect 공급자를 설정하는 데 사용되는 공개 및 개인 RSA 키 파일을 생성합니다.

```
$ ccoctl aws create-key-pair
```

출력 예:

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

여기서 **serviceaccount-signer.private** 및 **serviceaccount-signer.public**은 생성된 키 파일입니다.

또한 이 명령은 /< **path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** 에 설치하는 동안 클러스터가 필요한 개인 키를 생성합니다.

2. AWS에서 OpenID Connect ID 공급자 및 S3 버킷을 생성합니다.

```
$ ccoctl aws create-identity-provider \
--name=<name> \
--region=<aws_region> \
--public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public
```

다음과 같습니다.

- **<name>**은 추적을 위해 생성된 클라우드 리소스에 태그를 지정하는 데 사용되는 이름입니다.
- **<AWS-region** >은 클라우드 리소스가 생성될 AWS 리전입니다.
- **<path_to_ccoctl_output_dir** >은 **ccoctl aws create-key-pair** 명령이 생성된 공개 키 파일의 경로입니다.

출력 예:

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at
.well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json
updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::
<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

여기서 **02-openid-configuration**은 검색 문서이고 **03-keys.json**은 JSON 웹 키 집합 파일입니다.

또한 이 명령은 **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** 에 YAML 구성 파일을 생성합니다. 이 파일은 AWS IAM ID 공급자가 토큰을 신뢰하도록 클러스터가 생성하는 서비스 계정 토큰의 발급자 URL 필드를 설정합니다.

3. 클러스터의 각 구성 요소에 대한 IAM 역할을 생성합니다.

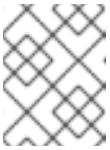
   a. OpenShift Container Platform 릴리스 이미지에서 **CredentialsRequest** 오브젝트 목록을 추출합니다.

```
$ oc adm release extract \
--credentials-requests \
--cloud=aws \
--to=<path_to_directory_with_list_of_credentials_requests>/credrequests (1)
--from=quay.io/<path_to>/ocp-release:<version>
```

   (1) **credrequests** 는 **CredentialsRequest** 오브젝트 목록이 저장되는 디렉터리입니다. 이 명령은 디렉터리가 없는 경우 해당 디렉터리를 생성합니다.

   b. **ccoctl** 툴을 사용하여 **redrequests** 디렉터리의 모든 **CredentialsRequest** 오브젝트를 처리합니다.

```
$ ccoctl aws create-iam-roles \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \
--identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```



참고

GovCloud와 같은 대체 IAM API 끝점을 사용하는 AWS 환경의 경우 **--region** 매개변수를 사용하여 리전을 지정해야 합니다.

각 **CredentialsRequest** 오브젝트에 대해 **ccoctl** 은 지정된 OIDC ID 공급자와 연결된 신뢰 정책과 OpenShift Container Platform 릴리스 이미지의 각 **CredentialsRequest** 오브젝트에 정의된 권한 정책을 사용하여 IAM 역할을 생성합니다.

검증

- OpenShift Container Platform 보안이 생성되었는지 확인하려면 < **path_to_ccoctl_output_dir>/manifests** 디렉터리의 파일을 나열합니다.

```
$ ll <path_to_ccoctl_output_dir>/manifests
```

출력 예:

```
total 24
-rw-------. 1 <user> <user> 161 Apr 13 11:42 cluster-authentication-02-config.yaml
-rw-------. 1 <user> <user> 379 Apr 13 11:59 openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
-rw-------. 1 <user> <user> 353 Apr 13 11:59 openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
-rw-------. 1 <user> <user> 355 Apr 13 11:59 openshift-image-registry-installer-cloud-credentials-credentials.yaml
-rw-------. 1 <user> <user> 339 Apr 13 11:59 openshift-ingress-operator-cloud-credentials-credentials.yaml
-rw-------. 1 <user> <user> 337 Apr 13 11:59 openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS를 쿼리하여 IAM 역할이 생성되었는지 확인할 수 있습니다. 자세한 내용은 IAM 역할 나열에 대한 AWS 설명서를 참조하십시오.

18.5.2.2.2. 단일 명령으로 AWS 리소스 생성

AWS 리소스를 수정하기 전에 **ccoctl** 툴에서 생성하는 JSON 파일을 검토할 필요가 없으며, **ccoctl** 툴에서 AWS 리소스를 생성하는 데 사용하는 프로세스가 조직의 요구 사항을 자동으로 충족하는 경우 **ccoctl aws create-all** 명령을 사용하여 AWS 리소스 생성을 자동화할 수 있습니다.

그렇지 않으면 AWS 리소스를 개별적으로 생성할 수 있습니다.

참고

기본적으로 **ccoctl**은 명령이 실행되는 디렉터리에 오브젝트를 생성합니다. 다른 디렉터리에 오브젝트를 생성하려면 **--output-dir** 플래그를 사용합니다. 이 절차에서는 & **lt;path_to_ccoctl_output_dir** >을 사용하여 이 디렉터리를 참조합니다.

사전 요구 사항

- **ccoctl** 바이너리를 추출하고 준비합니다.

절차

1. OpenShift Container Platform 릴리스 이미지에서 **CredentialsRequest** 오브젝트 목록을 추출합니다.

```
$ oc adm release extract \
--credentials-requests \
--cloud=aws \
--to=<path_to_directory_with_list_of_credentials_requests>/credrequests \ 1
--from=quay.io/<path_to>/ocp-release:<version>
```

1 **credrequests** 는 **CredentialsRequest** 오브젝트 목록이 저장되는 디렉터리입니다. 이 명령은 디렉터리가 없는 경우 해당 디렉터리를 생성합니다.

2. **ccoctl** 툴을 사용하여 **redrequests** 디렉터리의 모든 **CredentialsRequest** 오브젝트를 처리합니다.

```
$ ccoctl aws create-all \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests
```

검증

- OpenShift Container Platform 보안이 생성되었는지 확인하려면 < **path_to_ccoctl_output_dir>/manifests** 디렉터리의 파일을 나열합니다.

```
$ ll <path_to_ccoctl_output_dir>/manifests
```

출력 예:

```
total 24
-rw-------. 1 <user> <user> 161 Apr 13 11:42 cluster-authentication-02-config.yaml
-rw-------. 1 <user> <user> 379 Apr 13 11:59 openshift-cloud-credential-operator-cloud-
credential-operator-iam-ro-creds-credentials.yaml
-rw-------. 1 <user> <user> 353 Apr 13 11:59 openshift-cluster-csi-drivers-ebs-cloud-
credentials-credentials.yaml
-rw-------. 1 <user> <user> 355 Apr 13 11:59 openshift-image-registry-installer-cloud-
credentials-credentials.yaml
-rw-------. 1 <user> <user> 339 Apr 13 11:59 openshift-ingress-operator-cloud-credentials-
credentials.yaml
-rw-------. 1 <user> <user> 337 Apr 13 11:59 openshift-machine-api-aws-cloud-credentials-
credentials.yaml
```

AWS를 쿼리하여 IAM 역할이 생성되었는지 확인할 수 있습니다. 자세한 내용은 IAM 역할 나열에 대한 AWS 설명서를 참조하십시오.

18.5.2.3. 설치 프로그램 실행

사전 요구 사항

- OpenShift Container Platform 릴리스 이미지를 가져옵니다.

절차

1. 설치 프로그램이 포함된 디렉터리로 변경하고 **install-config.yaml** 파일을 생성합니다.

```
$ openshift-install create install-config --dir <installation_directory>
```

여기서 **<installation_directory>** 는 설치 프로그램이 파일을 생성하는 디렉터리입니다.

2. **install-config.yaml** 구성 파일을 편집하여 **credentialsMode** 매개 변수가 **Manual**로 설정되도록 합니다.

install-config.yaml 설정 파일 예

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual 1
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

1 이 행은 **credentialsMode** 매개변수를 **Manual**로 설정하기 위해 추가됩니다.

3. 필요한 OpenShift Container Platform 설치 매니페스트를 생성합니다.

```
$ openshift-install create manifests
```

4. **ccoctl**이 생성한 매니페스트를 설치 프로그램이 생성한 매니페스트 디렉터리에 복사합니다.

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. **ccoctl**이 **tls** 디렉터리에 생성된 개인 키를 설치 디렉터리에 복사합니다.

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6. OpenShift Container Platform 설치 프로그램을 실행합니다.

```
$ ./openshift-install create cluster
```

18.5.2.4. 설치 확인

1. OpenShift Container Platform 클러스터에 연결합니다.
2. 클러스터에 **root** 인증 정보가 없는지 확인합니다.

```
$ oc get secrets -n kube-system aws-creds
```

출력은 다음과 유사해야 합니다.

```
Error from server (NotFound): secrets "aws-creds" not found
```

3. 구성 요소가 CCO에서 생성한 인증 정보를 사용하는 대신 시크릿 매니페스트에 지정된 IAM 역할을 가정하는지 확인합니다.

이미지 레지스트리 Operator가 있는 명령 예

```
$ oc get secrets -n openshift-image-registry installer-cloud-credentials -o json | jq -r
.data.credentials | base64 --decode
```

출력에 구성 요소에서 사용하는 역할 및 웹 ID 토큰이 표시되고 다음과 유사해야 합니다.

이미지 레지스트리 Operator가 있는 출력 예

```
[default]
role_arn = arn:aws:iam::123456789:role/openshift-image-registry-installer-cloud-credentials
web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
```