OpenShift Container Platform 4.8

认证和授权

为用户和服务配置用户身份验证和访问控制

Last Updated: 2023-04-19

OpenShift Container Platform 4.8 认证和授权

为用户和服务配置用户身份验证和访问控制

法律通告

摘要

本文档提供在 OpenShift Container Platform 中定义身份提供程序的说明。它还讨论如何配置基于角色的访问控制来保护集群的安全。

目录

第 1 章 身份验证和授权概述

1.1. OPENSHIFT CONTAINER PLATFORM 身份验证和授权的常见术语表

此术语表定义了 OpenShift Container Platform 身份验证和授权中使用的常用术语。

身份验证

身份验证决定了对 OpenShift Container Platform 集群的访问，并确保只有经过身份验证的用户可以访问 OpenShift Container Platform 集群。

授权

授权决定识别的用户是否有权限来执行所请求的操作。

bearer 令牌

bearer 令牌用于通过标头 **Authorization: Bearer <token>** 向 API 进行身份验证。

Cloud Credential Operator

Cloud Credential Operator（CCO） 将云供应商凭证作为自定义资源定义 (CRD) 进行管理。

配置映射

配置映射提供将配置数据注入 pod 的方法。您可以在类型为 **ConfigMap** 的卷中引用存储在配置映射中的数据。在 pod 中运行的应用程序可以使用这个数据。

containers

包括软件及其所有依赖项的轻量级和可执行镜像。由于容器虚拟化操作系统，因此您可以在数据中心、公共云或私有云或本地主机中运行容器。

自定义资源 (CR)

CR 是 Kubernetes API 的扩展。

group

组是一组用户。组可用于一次性向多个用户授予权限。

HTPasswd

htpasswd 更新存储 HTTP 用户验证的用户名和密码的文件。

Keystone

Keystone 是一个 Red Hat OpenStack Platform (RHOSP)项目，提供身份、令牌、目录和策略服务。

轻量级目录访问协议 (LDAP)

LDAP 是查询用户信息的协议。

手动模式

在手动模式中，用户管理云凭证而不是 Cloud Credential Operator（CCO）。

Mint 模式

Mint 模式是 Cloud Credential Operator（CCO）的默认和推荐的最佳实践设置，用于支持它的平台。在这个模式下，CCO 使用提供的管理员级云凭证为集群中组件创建新凭证，且只具有所需的特定权限。

namespace

命名空间隔离所有进程可见的特定系统资源。在一个命名空间中，只有属于该命名空间的进程才能看到这些资源。

node

节点是 OpenShift Container Platform 集群中的 worker 机器。节点是虚拟机 (VM) 或物理计算机。

OAuth 客户端

OAuth 客户端用于获取 bearer 令牌。

OAuth 服务器

OpenShift Container Platform control plane 包含内置的 OAuth 服务器，用于决定用户身份来自配置的身份提供程序并创建访问令牌。

OpenID Connect

OpenID Connect 是一种协议，用于验证用户使用单点登录(SSO)来访问使用 OpenID 提供程序的站点。

passthrough 模式

在 passthrough 模式中，Cloud Credential Operator（CCO）将提供的云凭证传递给请求云凭证的组件。

pod

pod 是 Kubernetes 中的最小逻辑单元。pod 由一个或多个容器组成，可在 worker 节点上运行。

常规用户

首次登录时或通过 API 自动创建的用户。

请求标头(Request header)

请求标头是一个 HTTP 标头，用于提供有关 HTTP 请求上下文的信息，以便服务器可以跟踪请求的响应。

基于角色的访问控制 (RBAC)

重要的安全控制，以确保集群用户和工作负载只能访问执行其角色所需的资源。

服务帐户

服务帐户供集群组件或应用程序使用。

系统用户

安装集群时自动创建的用户。

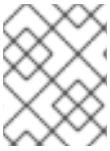
users

用户是可以向 API 发出请求的实体。

1.2. 关于 OPENSHIFT CONTAINER PLATFORM 中的身份验证

为了控制对 OpenShift Container Platform 集群的访问，集群管理员可以配置 用户身份验证，并确保只有批准的用户访问集群。

要与 OpenShift Container Platform 集群交互，用户必须首先以某种方式向 OpenShift Container Platform API 进行身份验证。您可以通过向 OpenShift Container Platform API 的请求中提供 OAuth 访问令牌或 X.509 客户端证书 进行验证。

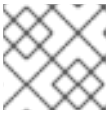


注意

如果您没有提供有效的访问令牌或证书，则您的请求会未经身份验证，您会收到 HTTP 401 错误。

管理员可以通过以下任务配置身份验证：

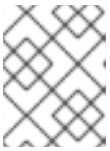
- 配置身份提供程序：您可以在 OpenShift Container Platform 中定义任何支持的身份提供程序，并将其添加到集群中。
- 配置内部 OAuth 服务器 ：OpenShift Container Platform control plane 包含内置的 OAuth 服务器，用于决定用户身份来自配置的身份提供程序并创建访问令牌。您可以配置令牌持续时间和不活跃超时。



注意

用户可以查看和管理归其拥有的 OAuth 令牌。

- 注册 OAuth 客户端：OpenShift Container Platform 包括几个 默认 OAuth 客户端。您可以注册和配置其他 OAuth 客户端。



注意

当用户为 OAuth 令牌发送请求时，必须指定接收和使用令牌的默认或自定义 OAuth 客户端。

- 使用 Cloud Credentials Operator 管理云供应商凭证：集群组件使用云供应商凭证来获取执行集群相关任务所需的权限。
- 模拟系统管理员用户：您可以通过模拟系统管理员用户来授予用户集群管理员权限。

1.3. 关于 OPENSHIFT CONTAINER PLATFORM 中的授权

授权涉及确定用户是否有权限来执行请求的操作。

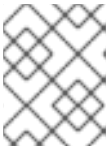
管理员可以定义权限，并使用 RBAC 对象（如规则、角色和绑定）将它们分配给用户。要了解授权在 OpenShift Container Platform 中的工作方式，请参阅评估授权。

您还可以通过项目和命名空间来控制对 OpenShift Container Platform 集群的访问。

除了控制用户对集群的访问外，您还可以控制 Pod 可以执行的操作，以及它可使用 安全性上下文约束 (SCC) 访问的资源。

您可以通过以下任务管理 OpenShift Container Platform 的授权：

- 查看 本地和集群 角色和绑定.
- 创建 本地角色 并将其分配给用户或组。
- 创建集群角色并将其分配给用户或组：OpenShift Container Platform 包括了一组默认的集群角色。您可以创建额外的 集群角色，并将它们添加到用户或组中。
- 创建 cluster-admin 用户：默认情况下，您的集群只有一个集群管理员，名为 **kubeadmin**。您可以创建另一个集群管理员。在创建集群管理员前，请确定您配置了身份提供程序。



注意

在创建了 cluster admin 用户后，删除现有的 kubeadmin 用户 来提高集群安全性。

- 创建服务帐户： 服务帐户 为控制 API 访问提供了灵活的方式，而无需共享常规用户的凭证。用户可以创建并在应用中使用服务帐户，也可以作为 OAuth 客户端。
- 有范围令牌：有范围令牌是一种令牌，指定只能执行特定操作的特定用户。您可以创建有范围令牌，将某些权限委派给其他用户或服务帐户。
- 同步 LDAP 组： 您可以通过将 存储在 LDAP 服务器中的组与 OpenShift Container Platform 用户组同步，以从一个单一的地方管理用户组。

第 2 章 了解身份验证

用户若要与 OpenShift Container Platform 交互，必须先进行集群的身份验证。身份验证层识别与 OpenShift Container Platform API 请求关联的用户。然后，授权层使用有关请求用户的信息来确定是否允许该请求。

作为管理员，您可以为 OpenShift Container Platform 配置身份验证。

2.1. 用户

OpenShift Container Platform 中的*用户*是可以向 OpenShift Container Platform API 发出请求的实体。OpenShift Container Platform **User** 对象代表操作者，通过向它们或所在的组添加角色为其授予系统中的权限。通常，这代表与 OpenShift Container Platform 交互的开发人员或管理员的帐户。

可能存在的用户类型有几种：

用户类型	描述
常规用户	这是大多数交互式 OpenShift Container Platform 用户的类型。常规用户于第一次登录时在系统中自动创建，或者也可通过 API 创建。常规用户通过 **User** 对象表示。例如，**joe alice**
系统用户	许多系统用户在基础架构定义时自动创建，主要用于使基础架构与 API 安全地交互。这包括集群管理员（有权访问一切资源）、特定于一个节点的用户、供路由器和 registry 使用的用户，以及一些其他用户。最后，还有一种 **anonymous** 系统用户，默认供未经身份验证的请求使用。例如：**system:admin system:openshift-registry system:node:node1.example.com**
服务帐户	服务帐户是与项目关联的特殊系统用户；有些是首次创建项目时自动创建的，而项目管理员则可为访问项目的内容创建更多的服务帐户。服务帐户通过 **ServiceAccount** 对象表示。例如：**system:serviceaccount:default:deployer system:serviceaccount:foo:builder**

每一用户必须通过某种形式的身份验证才能访问 OpenShift Container Platform。无身份验证或身份验证无效的 API 请求会被看作为由 **anonymous** 系统用户发出的请求。经过身份验证后，策略决定用户被授权执行的操作。

2.2. 组

用户可以分配到一个或多个*组*中，每个组代表特定的用户集合。在管理授权策略时，可使用组同时为多个用户授予权限，例如允许访问一个项目中的多个对象，而不必单独授予用户权限。

除了明确定义的组外，还有系统组或*虚拟组*，它们由集群自动置备。

以下列出了最重要的默认虚拟组：

虚拟组	描述
system:authenticated	自动与所有经过身份验证的用户关联。

虚拟组	描述
system:authenticated:oauth	自动与所有使用 OAuth 访问令牌经过身份验证的用户关联。
system:unauthenticated	自动与所有未经身份验证的用户关联。

2.3. API 身份验证

对 OpenShift Container Platform API 的请求通过以下方式进行身份验证：

OAuth 访问令牌

- 使用 ***<namespace_route>*/oauth/authorize** 和 ***<namespace_route>*/oauth/token** 端点，从 OpenShift Container Platform OAuth 服务器获取。
- 作为 **Authorization: Bearer...** 标头形式发送。
- 以 **base64url.bearer.authorization.k8s.io.<base64url-encoded-token>** 形式，作为 websocket 请求的 websocket 子协议标头发送。

X.509 客户端证书

- 需要与 API 服务器的 HTTPS 连接。
- 由 API 服务器针对可信证书颁发机构捆绑包进行验证。
- API 服务器创建证书并分发到控制器，以对自身进行身份验证。

任何具有无效访问令牌或无效证书的请求都会被身份验证层以 **401** 错误形式拒绝。

如果没有出示访问令牌或证书，身份验证层会将 **system:anonymous** 虚拟用户和 **system:unauthenticated** 虚拟组分配给请求。这使得授权层能够决定匿名用户可以发出哪些（如有）请求。

2.3.1. OpenShift Container Platform OAuth 服务器

OpenShift Container Platform master 包含内置的 OAuth 服务器。用户获取 OAuth 访问令牌来对自身进行 API 身份验证。

有人请求新的 OAuth 令牌时，OAuth 服务器使用配置的身份提供程序来确定提出请求的人的身份。

然后，它会确定该身份所映射到的用户，为该用户创建一个访问令牌，再返回要使用的令牌。

2.3.1.1. OAuth 令牌请求

每个对 OAuth 令牌的请求都必须指定要接收和使用令牌的 OAuth 客户端。启动 OpenShift Container Platform API 时会自动创建以下 OAuth 客户端：

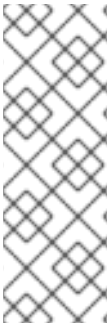
OAuth 客户端	使用方法
openshift-browser-client	使用可处理交互式登录的用户代理，在 **<namespace_route>/oauth/token/request** 请求令牌。[1]
openshift-challenging-client	使用可处理 **WWW-Authenticate** 质询的用户代理来请求令牌。

1. **<namespace_route>** 是指命名空间路由。运行以下命令可以找到：

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

所有对 OAuth 令牌的请求都包括对 **<namespace_route>/oauth/authorize** 的请求。大部分身份验证集成都会在这个端点前放置一个身份验证代理，或者将 OpenShift Container Platform 配置为针对后备身份提供程序验证凭证。对 **<namespace_route>/oauth/authorize** 的请求可能来自不能显示交互式登录页面的用户代理，如 CLI。因此，除了交互式登录流程外，OpenShift Container Platform 也支持使用 **WWW-Authenticate** 质询进行验证。

如果在 **<namespace_route>/oauth/authorize** 端点前面放置身份验证代理，它会向未经身份验证的非浏览器用户代理发送 **WWW-Authenticate** 质询，而不显示交互式登录页面或重定向到交互式登录流程。



注意

为防止浏览器客户端遭受跨站请求伪造 (CSRF) 攻击，当请求中存在 **X-CSRF-Token** 标头时，仅发送基本身份验证质询。希望接收基本 **WWW-Authenticate** 质询的客户端必须将此标头设置为非空值。

如果身份验证代理不支持 **WWW-Authenticate** 质询，或者如果 OpenShift Container Platform 配置为使用不支持 WWW-Authenticate 质询的身份提供程序，则必须使用浏览器从 **<namespace_route>/oauth/token/request** 手动获取令牌。

2.3.1.2. API 模仿

您可以配置对 OpenShift Container Platform API 的请求，使其表现为像是源自于另一用户。如需更多信息，请参阅 Kubernetes 文档中的用户模仿。

2.3.1.3. Prometheus 身份验证指标

OpenShift Container Platform 在身份验证尝试过程中捕获以下 Prometheus 系统指标：

- **openshift_auth_basic_password_count** 统计 **oc login** 用户名和密码的尝试次数。
- **openshift_auth_basic_password_count_result** 按照结果（**success** 或 **error**）统计 **oc login** 用户名和密码的尝试次数。
- **openshift_auth_form_password_count** 统计 web 控制台登录尝试次数。
- **openshift_auth_form_password_count_result** 按照结果（**success** 或 **error**）统计 web 控制台登录尝试次数。
- **openshift_auth_password_total** 统计 **oc login** 和 web 控制台登录尝试总次数。

第 3 章 配置内部 OAUTH 服务器

3.1. OPENSHIFT CONTAINER PLATFORM OAUTH 服务器

OpenShift Container Platform master 包含内置的 OAuth 服务器。用户获取 OAuth 访问令牌来对自身进行 API 身份验证。

有人请求新的 OAuth 令牌时，OAuth 服务器使用配置的身份提供程序来确定提出请求的人的身份。

然后，它会确定该身份所映射到的用户，为该用户创建一个访问令牌，再返回要使用的令牌。

3.2. OAUTH 令牌请求流量和响应

OAuth 服务器支持标准的授权代码授权和隐式授权 OAuth 授权流。

在使用隐式授权流 (**response_type=token**) 以及配置为请求 **WWW-Authenticate** 质询（如 **openshift-challenging-client**）的 client_id 以请求 OAuth 令牌时，可能来自 **/oauth/authorize** 的服务器响应及它们的处理方式如下方所列：

状态	内容	客户端响应
302	**Location** 标头包含 URL 片段中的 **access_token** 参数（RFC 6749 4.2.2 部分）	使用 **access_token** 值作为 OAuth 令牌。
302	**Location** 标头包含 **error** 查询参数（RFC 6749 4.1.2.1 部分）	失败，或向用户显示 **error**（使用可选的 **error_description**）查询值
302	其他 **Location** 标头	接续重定向操作，并使用这些规则处理结果
401	**WWW-Authenticate** 标头存在	在识别了类型时（如 **Basic** 和 **Negotiate** 等）响应质询，重新提交请求，再使用这些规则处理结果。
401	没有 **WWW-Authenticate** 标头	无法进行质询身份验证。失败，并显示响应正文（可能包含用于获取 OAuth 令牌的链接或备用方法详情）。
其他	其他	失败，或可向用户显示响应正文。

3.3. 内部 OAUTH 服务器选项

内部 OAuth 服务器可使用几个配置选项。

3.3.1. OAuth 令牌期间选项

内部 OAuth 服务器生成两种令牌：

令牌	描述
访问令牌	存在时间较长的令牌，用于授权对 API 的访问。
授权代码	存在时间较短的令牌，仅用于交换访问令牌。

您可以为两种类型的令牌配置默认的期间。若有需要，可使用 **OAuthClient** 对象定义覆盖访问令牌的期间。

3.3.2. OAuth 授权选项

当 OAuth 服务器收到用户之前没有授予权限的客户端的令牌请求时，OAuth 服务器采取的操作取决于 OAuth 客户端的授权策略。

请求令牌的 OAuth 客户端必须提供自己的授权策略。

您可以应用以下默认方法：

授权选项	描述
auto	自动批准授权并重试请求。
prompt	提示用户批准或拒绝授权。

3.4. 配置内部 OAUTH 服务器的令牌期间

您可以配置内部 OAuth 服务器令牌期间的默认选项。

重要

默认情况下，令牌仅在 24 小时内有效。现有会话会在此时间过后到期。

如果默认时间不足，可以使用以下步骤进行修改。

流程

1. 创建一个包含令牌期间选项的配置文件。以下文件将此周期设为 48 小时，两倍于默认值。

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

(1) 设置 **accessTokenMaxAgeSeconds** 以控制访问令牌的生命周期。默认生命周期为 24 小时或 86400 秒。此属性不可为负数。如果设置为零，则使用默认的生命周期。

2. 应用新配置文件：

注意

由于您更新现有的 OAuth 服务器，因此必须使用 **oc apply** 命令来应用更改。

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

3. 确认更改已生效：

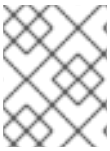
```
$ oc describe oauth.config.openshift.io/cluster
```

输**出示例**

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

3.5. 为内部 OAUTH 服务器配置令牌不活跃超时

您可以将 OAuth 令牌配置为在一组不活跃时间后过期。默认情况下，不会设置令牌不活跃超时。



注意

如果您的 OAuth 客户端中也配置了令牌不活动超时，则该值会覆盖内部 OAuth 服务器配置中设置的超时。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已经配置了一个身份提供程序（IDP）。

流程

1. 更新 **OAuth** 配置，以设置令牌不活跃超时。
   a. 编辑 **OAuth** 对象：

```
$ oc edit oauth cluster
```

添加 **spec.tokenConfig.accessTokenInactivityTimeout** 字段并设置超时值：

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
...
spec:
  tokenConfig:
    accessTokenInactivityTimeout: 400s (1)
```

```
-
```

1. 设定有适当单位的值，例如 **400s** 代表 400 秒，或 **30m** 代表 30 分钟。允许的超时最小值为 **300s**。

b. 保存文件以使改变生效。

2. 检查 OAuth 服务器 pod 是否已重启：

```
$ oc get clusteroperators authentication
```

在 **PROGRESSING** 列为 **False** 前不要继续进入下一步，如下所示：

输出示例

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

3. 检查是否已推出 Kubernetes API 服务器 pod 的新修订版本。这需要几分钟时间。

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

在 **PROGRESSING** 列为 **False** 前不要继续进入下一步，如下所示：

输出示例

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

如果 **PROGRESSING** 显示为 **True**，请等待几分钟后再试一次。

验证

1. 使用来自您的 IDP 的身份登录到集群。
2. 执行命令并确认它是否成功。
3. 等待的时间比配置的超时时间长而无需使用身份。在这个示例中，等待的时间超过 400 秒。
4. 尝试从同一身份的会话中执行命令。
   这个命令会失败，因为令牌应该因为不活跃的时间超过配置的超时时间而过期。

输出示例

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

3.6. OAUTH 服务器元数据

在 OpenShift Container Platform 中运行的应用程序可能需要发现有关内置 OAuth 服务器的信息。例如，它们可能需要发现 **<namespace_route>** 的哪个地址没有手动配置。为此，OpenShift Container Platform 实施了 IETF OAuth 2.0 授权服务器元数据草案规范。

因此，集群中运行的任何应用程序都可以向 ***https://openshift.default.svc/.well-known/oauth-authorization-server*** 发出 **GET** 请求来获取以下信息：

```
{
 "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. 授权服务器的签发者标识符，它是使用 **https** 方案且没有查询或分段组件的 URL。这是包含授权服务器有关信息的 **.well-known** RFC 5785 资源的发布位置。
2. 授权服务器的授权端点的 URL。参见 RFC 6749。
3. 授权服务器的令牌端点的 URL。参见 RFC 6749。
4. 包含此授权服务器支持的 OAuth 2.0 RFC 6749 范围值列表的 JSON 数组。请注意，并非所有支持的范围值都会公告。
5. 包含此授权服务器支持的 OAuth 2.0 **response_type** 值列表的 JSON 数组。使用的数组值与 **response_types** 参数（根据 RFC 7591 中"OAuth 2.0 Dynamic Client Registration Protocol"定义）使用的数组值相同。
6. 包含此授权服务器支持的 OAuth 2.0 授权类型值列表的 JSON 数组。使用的数组值与通过 **grant_types** 参数（根据 RFC 7591 中"**OAuth 2.0 Dynamic Client Registration Protocol**"定义）使用的数组值相同。
7. 包含此授权服务器支持的 PKCE RFC 7636 代码质询方法列表的 JSON 数组。**code_challenge_method** 参数中使用的代码质询方法值在 RFC 7636 第 4.3 节 中定义。有效的代码质询方法值是在 IANA **PKCE Code Challenge Methods** 注册表中注册的值。请参阅 IANA OAuth 参数。

3.7. OAUTH API 事件故障排除

在有些情况下，API 服务器会返回一个 **unexpected condition** 错误消息；若不直接访问 API 主日志，很难对此消息进行调试。该错误的基本原因被有意遮挡，以避免向未经身份验证的用户提供有关服务器状态的信息。

这些错误的子集与服务帐户 OAuth 配置问题有关。这些问题在非管理员用户可查看的事件中捕获。OAuth 期间遇到 **unexpected condition** 服务器错误时，可运行 **oc get events** 在 **ServiceAccount** 下查看这些事件。

以下示例警告缺少正确 OAuth 重定向 URI 的服务帐户：

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

输出示例

```
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>
```

运行 **oc describe sa/<service_account_name>** 可以报告与给定服务帐户名称关联的任何 OAuth 事件。

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

输出示例

```
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>
```

下方列出可能的事件错误：

无重定向 URI 注解或指定了无效的 URI

```
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>
```

指定了无效的路由

```
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>]
```

指定了无效的引用类型

```
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>]

缺少 SA 令牌

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

第 4 章 配置 OAUTH 客户端

在 OpenShift Container Platform 中默认创建多个 OAuth 客户端。您还可以注册和配置其他 OAuth 客户端。

4.1. 默认 OAUTH 客户端

启动 OpenShift Container Platform API 时会自动创建以下 OAuth 客户端：

OAuth 客户端	使用方法
openshift-browser-client	使用可处理交互式登录的用户代理，在 **<namespace_route>/oauth/token/request** 请求令牌。[1]
openshift-challenging-client	使用可处理 **WWW-Authenticate** 质询的用户代理来请求令牌。

1. **<namespace_route>** 是指命名空间路由。运行以下命令可以找到：

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

4.2. 注册其他 OAUTH 客户端

如果需要其他 OAuth 客户端来管理 OpenShift Container Platform 集群的身份验证，则可以注册一个。

流程

- 注册其他 OAuth 客户端：

```
$ 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 客户端的 **name** 用作提交对 **<namespace_route>/oauth/authorize** 和 **<namespace_route>/oauth/token** 的请求时的 **client_id** 参数。
2. **secret** 用作提交对 **<namespace_route>/oauth/token** 的请求时的 **client_secret** 参数。
3. 对 **<namespace_route>/oauth/authorize** 和 **<namespace_route>/oauth/token** 的请求中指定的 **redirect_uri** 参数必须等于 **redirectURIs** 参数值中所列的某一 URI 或以其为前缀。
4. **grantMethod** 用于确定当此客户端请求了令牌且还未被用户授予访问权限时应采取的操作。指定 **auto** 可自动批准授权并重试请求，或指定 **prompt** 以提示用户批准或拒绝授权。

4.3. 为 OAUTH 客户端配置令牌不活跃超时

在一组不活跃的时间后，您可以将 OAuth 客户端配置为使 OAuth 令牌过期。默认情况下，不会设置令牌不活跃超时。

注意

如果在内部 OAuth 服务器配置中也配置了令牌不活动超时，OAuth 客户端中设置的超时会覆盖该值。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已经配置了一个身份提供程序（IDP）。

流程

- 更新 **OAuthClient** 配置，以设置令牌不活跃超时。
  a. 编辑 **OAuthClient** 对象：

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

1 将 **<oauth_client>** 替换为要配置的 OAuth 客户端，如 **控制台**。

添加 **accessTokenInactivityTimeoutSeconds** 字段并设置您的超时值：

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

1 允许的最小超时值（以秒为单位）为 **300**。

  b. 保存文件以使改变生效。

验证

1. 使用来自您的 IDP 的身份登录到集群。确保使用您刚才配置的 OAuth 客户端。
2. 执行操作并验证它是否成功。
3. 等待的时间比配置的超时时间长而无需使用身份。在这个示例中，等待会超过 600 秒。
4. 尝试从同一身份的会话中执行一个操作。
   这个尝试会失败，因为令牌应该因为不活跃的时间超过配置的超时时间而过期。

4.4. 其他资源

- OAuthClient [oauth.openshift.io/v1]

第 5 章 管理用户拥有的 OAUTH 访问令牌

用户可查看其自身 OAuth 访问令牌，并删除不再需要的任何 OAuth 访问令牌。

5.1. 列出用户拥有的 OAUTH 访问令牌

您可以列出用户拥有的 OAuth 访问令牌。令牌名称并不敏感，它无法用于登录。

流程

- 列出所有用户拥有的 OAuth 访问令牌：

```
$ oc get useroauthaccesstokens
```

输**出示例**

```
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
```

- 列出特定 OAuth 客户端的用户拥有的 OAuth 访问令牌：

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

输**出示例**

```
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. 查看用户拥有的 OAUTH 访问令牌的详情

您可以查看用户拥有的 OAuth 访问令牌的详情。

流程

- 描述用户拥有的 OAuth 访问令牌的详情：

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

输**出示例**

```
Name:                 <token_name> 1
Namespace:
Labels:               <none>
```

```
Annotations:              <none>
API Version:              oauth.openshift.io/v1
Authorize Token:            sha256~Kscкkug-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. 令牌名称，即令牌的 sha256 哈希。令牌名称并不敏感，它无法用于登录。
2. 客户端名称，用于描述令牌来自哪里。
3. 从令牌创建到令牌过期间的时间（以秒为单位）。
4. 如果为 OAuth 服务器设置了令牌不活跃超时，则这是从创建到不再使用该令牌间的时间。
5. 此令牌的范围。
6. 与此令牌关联的用户名。

5.3. 删除用户拥有的 OAUTH 访问令牌

oc logout 命令只使活跃会话的 OAuth 令牌无效。您可以使用以下步骤删除不再需要的用户拥有的 OAuth 令牌。

从使用该令牌的所有会话中删除 OAuth 访问令牌日志。

流程

- 删除用户拥有的 OAuth 访问令牌：

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

输**出示例**

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

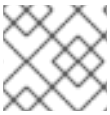
第 6 章 了解身份提供程序配置

OpenShift Container Platform master 包含内置的 OAuth 服务器。开发人员和管理员获取 OAuth 访问令牌，以完成自身的 API 身份验证。

作为管理员，您可以在安装集群后通过配置 OAuth 来指定身份提供程序。

6.1. 关于 OPENSHIFT CONTAINER PLATFORM 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。



注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

6.2. 支持的身份提供程序

您可以配置以下类型的身份提供程序：

用户身份提供程序	描述
htpasswd	配置 **htpasswd** 身份提供程序，针对使用 **htpasswd** 生成的文件验证用户名和密码。
Keystone	配置 **keystone** 身份提供程序，将 OpenShift Container Platform 集群与 Keystone 集成以启用共享身份验证，用配置的 OpenStack Keystone v3 服务器将用户存储到内部数据库中。
LDAP	配置 **ldap** 身份提供程序，使用简单绑定身份验证来针对 LDAPv3 服务器验证用户名和密码。
基本身份验证（Basic authentication）	配置 **basic-authentication** 身份提供程序，以便用户使用针对远程身份提供程序验证的凭证来登录 OpenShift Container Platform。基本身份验证是一种通用后端集成机制。
请求标头（Request header）	配置 **request-header** 身份提供程序，标识请求标头值中的用户，例如 **X-Remote-User**。它通常与设定请求标头值的身份验证代理一起使用。
Github 或 GitHub Enterprise	配置 **github** 身份提供程序，针对 GitHub 或 GitHub Enterprise 的 OAuth 身份验证服务器验证用户名和密码。
GitLab	配置 **gitlab** 身份提供程序，使用 GitLab.com 或任何其他 GitLab 实例作为身份提供程序。
Google	配置 **google** 身份提供程序，使用 Google 的 OpenID Connect 集成。
OpenID Connect	配置 **oidc** 身份提供程序，使用授权代码流与 OpenID Connect 身份提供程序集成。

定义了身份提供程序后，可以使用 RBAC 来定义并应用权限。

6.3. 移除 KUBEADMIN 用户

在定义了身份提供程序并创建新的 **cluster-admin** 用户后，您可以移除 **kubeadmin** 来提高集群安全性。

警告

如果在另一用户成为 **cluster-admin** 前按照这个步骤操作，则必须重新安装 OpenShift Container Platform。此命令无法撤销。

先决条件

- 必须至少配置了一个身份提供程序。
- 必须向用户添加了 **cluster-admin** 角色。
- 必须已经以管理员身份登录。

流程

- 移除 **kubeadmin** Secret：

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

6.4. 身份提供程序参数

以下是所有身份提供程序通用的参数：

参数	描述
name	此提供程序名称作为前缀放在提供程序用户名前，以此组成身份名称。

参数	描述
mappingMethod	定义在用户登录时如何将新身份映射到用户。输入以下值之一： **claim** 默认值。使用身份的首选用户名置备用户。如果具有该用户名的用户已映射到另一身份，则失败。 **lookup** 查找现有的身份、用户身份映射和用户，但不自动置备用户或身份。这允许集群管理员手动或使用外部流程设置身份和用户。使用此方法需要手动置备用户。 **generate** 使用身份的首选用户名置备用户。如果拥有首选用户名的用户已映射到现有的身份，则生成一个唯一用户名。例如：**myuser2**。此方法不应与需要在 OpenShift Container Platform 用户名和身份提供程序用户名（如 LDAP 组同步）之间完全匹配的外部流程一同使用。 **add** 使用身份的首选用户名置备用户。如果已存在具有该用户名的用户，此身份将映射到现有用户，添加到该用户的现有身份映射中。如果配置了多个身份提供程序并且它们标识同一组用户并映射到相同的用户名，则需要进行此操作。

注意

在添加或更改身份提供程序时，您可以通过把 **mappingMethod** 参数设置为 **add**，将新提供程序中的身份映射到现有的用户。

6.5. 身份提供程序 CR 示例

以下自定义资源 (CR) 显示用来配置身份提供程序的参数和默认值。本例使用 htpasswd 身份提供程序。

身份提供程序 CR 示例

```
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. 此提供程序名称作为前缀放在提供程序用户名前，以此组成身份名称。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 包含使用 **htpasswd** 生成的文件的现有 secret。

第 7 章 配置身份提供程序

7.1. 配置 HTPASSWD 身份提供程序

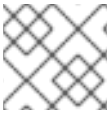
配置 **htpasswd** 身份提供程序，允许用户使用 htpasswd 文件中的凭证登录 OpenShift Container Platform。

要定义 htpasswd 身份提供程序，请执行以下任务：

1. 创建一个 **htpasswd** 文件来存储 用户和密码信息。
2. 创建一个 secret 来代表 **htpasswd** 文件。
3. 定义一个htpasswd 身份提供程序资源用于引用该 secret。
4. 将资源应用到 默认的 OAuth 配置，以添加身份提供程序。

7.1.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。



注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

7.1.2. 关于 htpasswd 身份验证

在 OpenShift Container Platform 中使用 htpasswd 身份验证允许您根据 htpasswd 文件识别用户。htpasswd 文件是一个平面文件，其中包含每个用户的用户名和密码。您可以使用 **htpasswd** 实用程序创建此文件。

7.1.3. 创建 htpasswd 文件

有关如何创建 htpasswd 文件的说明，请参见以下小节之一：

- 使用 Linux 创建 htpasswd 文件
- 使用 Windows 创建 htpasswd 文件

7.1.3.1. 使用 Linux 创建 htpasswd 文件

要使用 htpasswd 身份提供程序，您必须使用 **htpasswd** 生成一个包含集群用户名和密码的文件。

先决条件

- 能够访问 **htpasswd** 实用程序。在 Red Hat Enterprise Linux 上，安装 **httpd-tools** 软件包即可使用该实用程序。

流程

1. 创建或更新含有用户名和散列密码的平面文件：

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

该命令将生成散列版本的密码。

例如：

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

输出示例

```
Adding password for user user1
```

2. 继续向文件中添加或更新凭证：

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

7.1.3.2. 使用 Windows 创建 htpasswd 文件

要使用 htpasswd 身份提供程序，您必须使用 **htpasswd** 生成一个包含集群用户名和密码的文件。

先决条件

- 能够访问 **htpasswd.exe**。许多 Apache httpd 发行版本的 **\bin** 目录中均包含此文件。

流程

1. 创建或更新含有用户名和散列密码的平面文件：

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

该命令将生成散列版本的密码。

例如：

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

输出示例

```
Adding password for user user1
```

2. 继续向文件中添加或更新凭证：

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

7.1.4. 创建 htpasswd secret

要使用 htpasswd 身份提供程序，您必须定义一个包含 htpasswd 用户文件的 secret。

先决条件

- 创建 htpasswd 文件。

流程

- 创建包含 htpasswd 用户文件的 **Secret** 对象：

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

(1) 包含 **--from-file** 参数的用户文件的 secret 键必须命名为 **htpasswd**，如上述命令所示。

提示

您还可以应用以下 YAML 来创建 secret：

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

7.1.5. Sample htpasswd CR

以下自定义资源 (CR) 显示 htpasswd 身份提供程序的参数和可接受值。

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) 此提供程序名称作为前缀放在提供程序用户名前，以此组成身份名称。

(2) 控制如何在此提供程序的身份和 **User** 对象之间建立映射。

(3) 包含使用 **htpasswd** 生成的文件的现有 secret。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.1.6. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

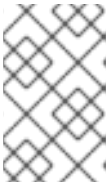
先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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



注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.1.7. 为 htpasswd 身份提供程序更新用户

您可以从现有的 htpasswd 身份提供程序中添加或删除用户。

先决条件

- 您已创建了包含 htpasswd 用户文件的 **Secret** 对象。这里假定它名为 **htpass-secret**。
- 您已配置了 htpasswd 身份提供程序。这里假定它名为 **my_htpasswd_provider**。
- 您可以使用 **htpasswd** 工具程序。在 Red Hat Enterprise Linux 上，安装 **httpd-tools** 软件包即可使用该实用程序。
- 您需要有集群管理员特权。

流程

1. 从 **htpass-secret Secret** 对象中检索 htpasswd 文件，并将该文件保存到您的文件系统中：

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

2. 从 **users.htpasswd** 文件中添加或删除用户。

   - 添加一个新用户：

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

     输**出示例**

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

   - 删除一个现有用户：

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

     输**出示例**

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

3. 将 **htpass-secret Secret** 对象替换为 **users.htpasswd** 文件中更新的用户：

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

   提示

   您还可以应用以下 YAML 来替换 secret：

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

4. 如果删除了一个或多个用户，您还需要为每个用户删除其现有资源。

   a. 删除 **User** 对象：

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

      输**出示例**

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

      请确认已删除了用户，否则如果用户的令牌还没有过期，则用户还可以继续使用其令牌。

   b. 删除用户的 **Identity** 对象：

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

输出示例

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

7.1.8. 使用 web 控制台配置身份提供程序

通过 web 控制台而非 CLI 配置身份提供程序 (IDP)。

先决条件

- 您必须以集群管理员身份登录到 web 控制台。

流程

1. 导航至 **Administration → Cluster Settings**。
2. 在 **Global Configuration** 选项卡下，点 **OAuth**。
3. 在 **Identity Providers** 部分中，从 **Add** 下拉菜单中选择您的身份提供程序。

注意

您可以通过 web 控制台来指定多个 IDP，而不会覆盖现有的 IDP。

7.2. 配置 KEYSTONE 身份提供程序

配置 **keystone** 身份提供程序，将 OpenShift Container Platform 集群与 Keystone 集成以启用共享身份验证，用配置的 OpenStack Keystone v3 服务器将用户存储到内部数据库中。此配置允许用户使用其 Keystone 凭证登录 OpenShift Container Platform。

7.2.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。

注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

7.2.2. 关于 Keystone 身份验证

Keystone 是一个提供身份、令牌、目录和策略服务的 OpenStack 项目。

您可以配置与 Keystone 的集成，以便 OpenShift Container Platform 的新用户基于 Keystone 用户名或者唯一 Keystone ID。使用这两种方法时，用户可以输入其 Keystone 用户名和密码进行登录。使 Keystone ID 上的 OpenShift Container Platform 用户更为安全，因为如果您删除 Keystone 用户并创建具有该用户名的新 Keystone 用户，新用户可能有权访问旧用户的资源。

7.2.3. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令，创建一个包含密钥和证书的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

```
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. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.2.5. Keystone CR 示例

以下自定义资源 (CR) 显示 Keystone 身份提供程序的参数和可接受值。

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. 此提供程序名称作为前缀放在提供程序用户名前，以此组成身份名称。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. Keystone 域名。在 Keystone 中，用户名是特定于域的。只支持一个域。
4. 用于连接到 Keystone 服务器的 URL（必需）。这必须使用 https。
5. 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform **ConfigMap** 的引用，以用于验证所配置 URL 的服务器证书。
6. 可选：对包含客户端证书的 OpenShift Container Platform **Secret** 对象的引用，该证书在向所配置的 URL 发出请求时出示。
7. 对包含客户端证书密钥的 OpenShift Container Platform **Secret** 对象的引用。指定了 **tlsClientCert** 时必需此项。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.2.6. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

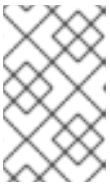
先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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



注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.3. 配置 LDAP 身份提供程序

配置 **ldap** 身份提供程序，使用简单绑定身份验证来针对 LDAPv3 服务器验证用户名和密码。

7.3.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。

注意

OpenShift Container Platform 用户名不能包括 **/**、**:** 和 **%**。

7.3.2. 关于 LDAP 身份验证

在身份验证过程中，搜索 LDAP 目录中与提供的用户名匹配的条目。如果找到一个唯一匹配项，则尝试使用该条目的可分辨名称 (DN) 以及提供的密码进行简单绑定。

执行下面这些步骤：

1. 通过将配置的 **url** 中的属性和过滤器与用户提供的用户名组合来生成搜索过滤器。
2. 使用生成的过滤器搜索目录。如果搜索返回的不是一个条目，则拒绝访问。
3. 尝试使用搜索所获条目的 DN 和用户提供的密码绑定到 LDAP 服务器。
4. 如果绑定失败，则拒绝访问。
5. 如果绑定成功，则将配置的属性用作身份、电子邮件地址、显示名称和首选用户名来构建一个身份。

配置的 **url** 是 RFC 2255 URL，指定要使用的 LDAP 主机和搜索参数。URL 的语法是：

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

在这个 URL 中：

URL 组件	描述
ldap	对于常规 LDAP，使用 **ldap** 字符串。对于安全 LDAP (LDAPS)，改为使用 **ldaps**。
host:port	LDAP 服务器的名称和端口。LDAP 默认为 **localhost:389**，LDAPS 则默认为 **localhost:636**。
basedn	所有搜索都应从中开始的目录分支的 DN。至少，这必须是目录树的顶端，但也可指定目录中的子树。
attribute	要搜索的属性。虽然 RFC 2255 允许使用逗号分隔属性列表，但无论提供多少个属性，都仅使用第一个属性。如果没有提供任何属性，则默认使用 **uid**。建议选择一个在您使用的子树中的所有条目间是唯一的属性。
scope	搜索的范围。可以是 **one** 或 **sub**。如果未提供范围，则默认使用 **sub** 范围。
filter	有效的 LDAP 搜索过滤器。如果未提供，则默认为 **(objectClass=*)**

在进行搜索时，属性、过滤器和提供的用户名会组合在一起，创建类似如下的搜索过滤器：

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

例如，可考虑如下 URL：

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

当客户端尝试使用用户名 **bob** 连接时，生成的搜索过滤器将为 **(&(enabled=true)(cn=bob))**。

如果 LDAP 目录需要身份验证才能搜索，请指定用于执行条目搜索的 **bindDN** 和 **bindPassword**。

7.3.3. 创建 LDAP secret

要使用身份提供程序，您必须定义一个包含 **bindPassword** 字段的 OpenShift Container Platform **Secret** 对象。

流程

- 创建包含 **bindPassword** 字段的 **Secret** 对象：

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

(1) 包含 **--from-literal** 参数的 bindPassword 的 secret 键必须称为 **bindPassword**。

提示

您还可以应用以下 YAML 来创建 secret：

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

7.3.4. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.3.5. LDAP CR 示例

以下自定义资源 (CR) 显示 LDAP 身份提供程序的参数和可接受值。

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. 此提供程序名称作为前缀放在返回的用户 ID 前，以此组成身份名称。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 用作身份的属性列表。使用第一个非空属性。至少需要一个属性。如果列出的属性都没有值，身份验证会失败。定义属性按原样检索，允许使用二进制值。
4. 用作电子邮件地址的属的列表。使用第一个非空属性。
5. 用作显示名称的属性列表。使用第一个非空属性。
6. 为此身份置备用户时用作首选用户名的属性列表。使用第一个非空属性。
7. 在搜索阶段用来绑定的可选 DN。如果定义了 **bindPassword**，则必须设置此项。
8. 对包含绑定密码的 OpenShift Container Platform **Secret** 对象的可选引用。如果定义了 **bindDN**，则必须设置此项。
9. 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform **ConfigMap** 的引用，以用于验证所配置 URL 的服务器证书。仅在 **insecure** 为 **false** 时使用。
10. 为 **true** 时，不会对服务器进行 TLS 连接。为 **false** 时，**ldaps://** URL 使用 TLS 进行连接，并且 **ldap://** URL 升级到 TLS。当使用 **ldaps://** URL 时，此项必须设为 **false**，因为这些 URL 始终会尝试使用 TLS 进行连接。
11. RFC 2255 URL，指定要使用的 LDAP 主机和搜索参数。

注意

要将用户列在 LDAP 集成的白名单中，请使用 **lookup** 映射方法。在允许从 LDAP 登录前，集群管理员必须为每个 LDAP **User** 创建一个 **Identity** 对象和用户对象。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.3.6. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

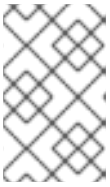
先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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



注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.4. 配置基本身份验证身份提供程序

配置 **basic-authentication** 身份提供程序，以便用户使用针对远程身份提供程序验证的凭证来登录 OpenShift Container Platform。基本身份验证是一种通用后端集成机制。

7.4.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。

注意

OpenShift Container Platform 用户名不能包括 **/**、**:** 和 **%**。

7.4.2. 关于基本身份验证

基本身份验证是一种通用后端集成机制，用户可以使用针对远程身份提供程序验证的凭证来登录 OpenShift Container Platform。

由于基本身份验证是通用的，因此您可以在高级身份验证配置中使用此身份提供程序。

重要

基本身份验证必须使用 HTTPS 连接到远程服务器，以防止遭受用户 ID 和密码嗅探以及中间人攻击。

配置了基本身份验证后，用户将其用户名和密码发送到 OpenShift Container Platform，然后通过提出服务器对服务器请求并将凭证作为基本身份验证标头来传递，针对远程服务器验证这些凭证。这要求用户在登录期间向 OpenShift Container Platform 发送凭证。

注意

这只适用于用户名/密码登录机制，并且 OpenShift Container Platform 必须能够向远程身份验证服务器发出网络请求。

针对受基本身份验证保护并返回 JSON 的远程 URL 验证用户名和密码。

401 响应表示身份验证失败。

非 **200** 状态或出现非空"error"键表示出现错误：

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

200 状态并带有 **sub**（subject）键则表示成功：

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

1 主体必须是经过身份验证的用户所特有的，而且必须不可修改。

成功响应可以有选择地提供附加数据，例如：

- 使用 **name** 键的显示名称。例如：

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

- 使用 **email** 键的电子邮件地址。例如：

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

- 使用 **preferred_username** 键的首选用户名。这可用在唯一不可改主体是数据库密钥或 UID 且存在更易读名称的情形中。为经过身份验证的身份置备 OpenShift Container Platform 用户时，这可用作提示。例如：

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

7.4.3. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令，创建一个包含密钥和证书的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

```
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. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.4.5. 基本身份验证 CR 示例

以下自定义资源（CR）显示基本身份验证身份提供程序的参数和可接受值。

基本身份验证 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. 此提供程序名称作为前缀放在返回的用户 ID 前，以此组成身份名称。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 接受基本身份验证标头中凭证的 URL。
4. 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform **ConfigMap** 的引用，以用于验证所配置 URL 的服务器证书。
5. 可选：对包含客户端证书的 OpenShift Container Platform **Secret** 对象的引用，该证书在向所配置的 URL 发出请求时出示。
6. 对包含客户端证书密钥的 OpenShift Container Platform **Secret** 对象的引用。指定了 **tlsClientCert** 时必需此项。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.4.6. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

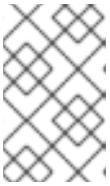
先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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



注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.4.7. 基本身份供应商的 Apache HTTPD 配置示例

OpenShift Container Platform 4 中的基本身份供应商 (IDP) 配置要求 IDP 服务器的 JSON 响应以获得成功和失败。您可以使用 Apache HTTPD 中的 CGI 脚本来达到此目的。本节提供示例。

/etc/httpd/conf.d/login.conf 示例

```
<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>
```

/var/www/cgi-bin/login.cgi 示例

```
#!/bin/bash
echo "Content-Type: application/json"
echo ""
```

```
echo '{"sub":"userid", "name":"'$REMOTE_USER'"}'
exit 0
```

/var/www/cgi-bin/fail.cgi 示例

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

7.4.7.1. 文件要求

以下是您在 Apache HTTPD Web 服务器中创建的文件要求：

- **login.cgi** 和 **fail.cgi** 必须可执行（**chmod +x**）。
- 如果启用了 SELinux，**login.cgi** 和 **fail.cgi** 需要有适当的 SELinux 上下文：**restorecon -RFv /var/www/cgi-bin**，或确保上下文是 **httpd_sys_script_exec_t**（运行 **ls -laZ**）。
- **login.cgi** 只有在用户根据 **Require and Auth** 项成功登陆时才执行。
- 如果用户无法登录，则执行**fail.cgi** ，并做出 **HTTP 401** 响应。

7.4.8. 基本身份验证故障排除

最常见的问题与后端服务器网络连接相关。要进行简单调试，请在 master 上运行 **curl** 命令。要测试成功的登录，请将以下示例命令中的 **<user>** 和 **<password>** 替换为有效的凭证。要测试无效的登录，请将它们替换为错误的凭证。

```
$ 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
```

成功响应

200 状态并带有 **sub**（subject）键则表示成功：

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

subject 必须是经过身份验证的用户所特有的，而且必须不可修改。

成功响应可以有选择地提供附加数据，例如：

- 使用 **name** 键的显示名称：

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

- 使用 **email** 键的电子邮件地址：

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

- 使用 **preferred_username** 键的首选用户名：

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

preferred_username 键可用在唯一不可改主体是数据库密钥或 UID 且存在更易读名称的情形中。为经过身份验证的身份置备 OpenShift Container Platform 用户时，这可用作提示。

失败的响应

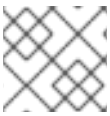
- **401** 响应表示身份验证失败。
- 非 **200** 状态或带有非空"error"键表示错误：**{"error":"Error message"}**

7.5. 配置请求标头身份提供程序

配置 **request-header** 身份提供程序，标识请求标头值中的用户，例如 **X-Remote-User**。它通常与设定请求标头值的身份验证代理一起使用。

7.5.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。



注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

7.5.2. 关于请求标头身份验证

请求标头身份提供程序从请求标头值标识用户，例如 **X-Remote-User**。它通常与设定请求标头值的身份验证代理一起使用。请求标头身份提供程序无法与其他使用直接密码登录的身份提供商结合使用 htpasswd、Keystone、LDAP 或基本身份验证。

注意

您还可以将请求标头身份提供程序用于高级配置，如由社区支持的 SAML 身份验证。请注意，红帽不支持这个解决方案。

用户使用此身份提供程序进行身份验证时，必须通过身份验证代理访问 **https://*<namespace_route>*/oauth/authorize**（及子路径）。要实现此目标，请将 OAuth 服务器配置为把 OAuth 令牌的未经身份验证的请求重定向到代理到 **https://*<namespace_route>*/oauth/authorize** 的代理端点。

重定向来自希望基于浏览器型登录流的客户端的未经身份验证请求：

- 将 **provider.loginURL** 参数设为身份验证代理 URL，该代理将验证交互式客户端并将其请求代理到 **https://*<namespace_route>*/oauth/authorize**。

重定向来自希望 **WWW-Authenticate** 质询的客户端的未经身份验证请求：

- 将 **provider.challengeURL** 参数设置为身份验证代理 URL，该代理将验证希望 **WWW-Authenticate** 质询的客户端并将其请求代理到 **https://*<namespace_route>*/oauth/authorize**。

provider.challengeURL 和 **provider.loginURL** 参数可以在 URL 的查询部分中包含以下令牌：

- **${url}** 替换为当前的 URL，进行转义以在查询参数中安全使用。

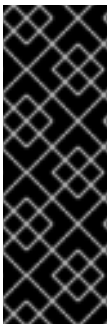
例如：**https://www.example.com/sso-login?then=${url}**

- **${query}** 替换为当前的查询字符串，不进行转义。
例如：**https://www.example.com/auth-proxy/oauth/authorize?${query}**

重要

自 OpenShift Container Platform 4.1 起，代理必须支持 mutual TLS。

7.5.2.1. Microsoft Windows 上的 SSPI 连接支持



重要

使用 Microsoft Windows 上的 SSPI 连接支持是技术预览功能。技术预览功能不包括在红帽生产服务级别协议（SLA）中，且其功能可能并不完善。因此，红帽不建议在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

如需红帽技术预览功能支持范围的更多信息，请参阅 https://access.redhat.com/support/offerings/techpreview/。

OpenShift CLI （**oc**）支持安全支持提供程序接口（SSPI），以允许 Microsft Windows 上的 SSO 流。如果您使用请求标头身份提供程序与支持 GSSAPI 的代理将 Active Directory 服务器连接到 OpenShift Container Platform，用户可以通过加入了域的 Microsoft Windows 计算机使用 **oc** 命令行界面来自动进行 OpenShift Container Platform 身份验证。

7.5.3. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.5.4. 请求标头 CR 示例

以下自定义资源 (CR) 显示请求标头身份提供程序的参数和可接受值。

请求标题 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. 此提供程序名称作为前缀放在请求标头中的用户名前，以此组成身份名称。

2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。

3. 可选：将未经身份验证的 **/oauth/authorize** 请求重定向到的 URL，它将身份验证基于浏览器的客户端并将其请求代理到 **https://*<namespace_route>*/oauth/authorize**。代理到 **https://*<namespace_route>*/oauth/authorize** 的 URL 必须以 **/authorize** 结尾（不含尾部斜杠），以及代理子路径，以便 OAuth 批准流可以正常工作。**${url}** 替换为当前的 URL，进行转义以在查询参数中安全使用。把 **${query}** 替换为当前的查询字符串。如果未定义此属性，则必须使用 **loginURL**。

4. 可选：将未经身份验证的 **/oauth/authorize** 请求重定向到的 URL，它将身份验证期望 **WWW-Authenticate** challenges 的客户端，然后将它们代理到 **https://*<namespace_route>*/oauth/authorize**。**${url}** 替换为当前的 URL，进行转义以在查询参数中安全使用。把 **${query}** 替换为当前的查询字符串。如果未定义此属性，则必须使用 **challengeURL**。

5. 对包含 PEM 编码证书捆绑包的 OpenShift Container Platform **ConfigMap** 的引用。用作信任定位符，以验证远程服务器出示的 TLS 证书。

重要

自 OpenShift Container Platform 4.1 起，此身份提供程序需要 **ca** 字段。这意味着您的代理必须支持 mutual TLS。

6 可选：通用名称 (**cn**) 的列表。如果设定，则必须出示带有指定列表中通用名称 (**cn**) 的有效客户端证书，然后才能检查请求标头中的用户名。如果为空，则允许任何通用名称。只能与 **ca** 结合使用。

7 按顺序查找用户身份的标头名称。第一个包含值的标头被用作身份。必需，不区分大小写。

8 按顺序查找电子邮件地址的标头名称。第一个包含值的标头被用作电子邮件地址。可选，不区分大小写。

9 按顺序查找显示名称的标头名称。第一个包含值的标头被用作显示名称。可选，不区分大小写。

10 按顺序查找首选用户名的标头名称（如果与通过 **headers** 中指定的标头确定的不可变身份不同）。在置备时，第一个包含值的标头用作首选用户名。可选，不区分大小写。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.5.5. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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

注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.5.6. 使用请求标头进行 Apache 验证的配置示例

本例使用请求标头身份提供程序为 OpenShift Container Platform 配置 Apache 验证代理服务器。

自定义代理配置
使用 **mod_auth_gssapi** 模块是使用请求标头身份提供程序配置 Apache 认证代理的流行方法，但这并不是必需的。如果满足以下要求，您可以轻松地使用其他代理：

- 阻断来自客户端请求的 **X-Remote-User** 标头以防止欺骗。
- 在 **RequestHeaderIdentityProvider** 配置中强制进行客户端证书验证 。
- 使用质询流来要求 **X-CSRF-Token** 标头为所有身份验证请求设置。
- 请确定只有 **/oauth/authorize** 端点和其子路径通过代理处理。重定向必须被重写，以便后端服务器可以将客户端发送到正确的位置。
- 代理到 **https://<namespace_route>/oauth/authorize** 的 URL 必须以 **/authorize** 结尾，且最后没有尾部斜杠。例如： **https://proxy.example.com/login-proxy/authorize?...** 必须代理到 **https://<namespace_route>/oauth/authorize?...**。
- 代理到 **https://<namespace_route>/oauth/authorize** 的 URL 的子路径必须代理至 **https://<namespace_route>/oauth/authorize** 的子路径。例如： **https://proxy.example.com/login-proxy/authorize/approve?...** 必须代理到 **https://<namespace_route>/oauth/authorize/approve?...**。

注意

https://<namespace_route> 地址是到 OAuth 服务器的路由，可通过运行 **oc get route -n openshift-authentication** 获取。

使用请求标头配置 Apache 身份验证
这个示例使用 **mod_auth_gssapi** 模块使用请求标头身份提供程序配置 Apache 验证代理。

先决条件

- 通过 Optional channel 获得 **mod_auth_gssapi** 模块。您必须在本地机器中安装以下软件包：
  - **httpd**
  - **mod_ssl**
  - **mod_session**
  - **apr-util-openssl**
  - **mod_auth_gssapi**
- 生成用于验证提交可信标头的请求的 CA。定义包含 CA 的 OpenShift Container Platform **ConfigMap** 对象。这可以通过运行以下命令完成：

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

(1) 证书颁发机构必须存储在 **ConfigMap** 的 **ca.crt** 键中。

提示

您还可以应用以下 YAML 来创建配置映射：

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

- 示例：为代理生成客户端证书您可以使用任何 x509 证书工具生成这个证书。客户端证书必须由您生成的 CA 签名，以验证提交可信标头的请求。
- 为身份提供程序创建自定义资源（CR）。

流程

此代理使用客户端证书连接到 OAuth 服务器，该服务器被配置为信任 **X-Remote-User** 标头。

1. 为 Apache 配置创建证书。您通过 **SSLProxyMachineCertificateFile** 参数值指定的证书是服务器验证代理时使用的代理客户端的证书。它必须使用 **TLS Web 客户端身份验证**作为扩展密钥类型。
2. 创建 Apache 配置文件。使用以下模板来提供所需设置和值：

重要

仔细检查模板的内容，并根据您的环境自定义其相应的内容。

```
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
```

注意

https://<namespace_route> 地址是到 OAuth 服务器的路由，可通过运行 **oc get route -n openshift-authentication** 获取。

3. 更新自定义资源 (CR) 中的 **identityProviders** 小节：

```
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. 验证配置：

   a. 通过提供正确的客户端证书和标头，确认您可以通过请求令牌来绕过代理：

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

   b. 通过在没有证书的情况下请求令牌，确认没有提供客户端证书的请求会失败：

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

   c. 确认 **challengeURL** 重定向已启用：

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

   复制 **challengeURL** 重定向，以用于下一步骤。

   d. 运行这个命令会显示一个带有 **WWW-Authenticate** 基本质询，协商质询或两个质询都有的 **401** 响应：

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

   e. 测试在使用 Kerberos ticket 和不使用 Kerberos ticket 的情况下登录到 OpenShift CLI（**oc**）：

i. 如果您使用 **kinit**生成了 Kerberos ticket，请将其销毁：

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

1 请确定提供 Kerberos 缓存的名称。

ii. 使用您的 Kerberos 凭证登录到 **oc**：

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

在提示符后输入您的 Kerberos 密码。

iii. 注销 **oc** 工具：

```
# oc logout
```

iv. 使用您的 Kerberos 凭证获得一个 ticket：

```
# kinit
```

在提示符后输入您的 Kerberos 用户名和密码。

v. 确认您可以登录到 **oc**：

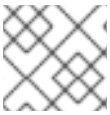
```
# oc login
```

如果配置正确，您会在不需要单独输入凭证的情况下成功登录。

7.6. 配置 GITHUB 或 GITHUB ENTERPRISE 身份提供程序

配置 **github** 身份提供程序，针对 GitHub 或 GitHub Enterprise 的 OAuth 身份验证服务器验证用户名和密码。OAuth 可以协助 OpenShift Container Platform 和 GitHub 或 GitHub Enterprise 之间的令牌交换流。

您可以使用 GitHub 集成来连接 GitHub 或 GitHub Enterprise。对于 GitHub Enterprise 集成，您必须提供实例的 **hostname**，并可选择提供要在服务器请求中使用的 **ca** 证书捆绑包。



注意

除非有所注明，否则以下步骤同时适用于 GitHub 和 GitHub Enterprise。

7.6.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。

注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

7.6.2. 关于 GitHub 身份验证

配置 GitHub 身份验证后，用户可使用 GitHub 凭证登录 OpenShift Container Platform。为防止具有任何 GitHub 用户 ID 的任何人登录 OpenShift Container Platform 集群，您可以将访问权利限制给仅属于特定 GitHub 组织的用户。

7.6.3. 注册 GitHub 应用程序

要将 GitHub 或 GitHub Enterprise 用作身份提供程序，您必须注册要使用的应用程序。

流程

1. 在 GitHub 上注册应用程序：
   - 对于 GitHub,点击 Settings → Developer settings → OAuth Apps → Register a new OAuth application。
   - 对于 GitHub Enterprise，前往 GitHub Enterprise 主页，然后点击 **Settings → Developer settings → Register a new application**。
2. 输入应用程序名称，如 **My OpenShift Install**。
3. 输入主页 URL，如 **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>**。
4. 可选：输入应用程序描述。
5. 输入授权回调 URL，其中 URL 末尾包含身份提供程序 **name**：

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

   例如：

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

6. 点击 **Register application**。GitHub 提供客户端 ID 和客户端 secret。您需要使用这些值来完成身份提供程序配置。

7.6.4. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令创建一个包含字符串的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

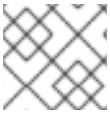
```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 您可以使用以下命令定义一个包含文件内容的 **Secret** 对象，如证书文件：

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

7.6.5. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。



注意

只有 GitHub Enterprise 需要此步骤。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.6.6. GitHub CR 示例

以下自定义资源 (CR) 显示 GitHub 身份提供程序的参数和可接受值。

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. 此提供程序名称作为前缀放在 GitHub 数字用户 ID 前，以此组成身份名称。它还可用来构建回调 URL。

2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。

3. 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform **ConfigMap** 的引用，以用于验证所配置 URL 的服务器证书。仅用于带有非公开信任的根证书的 GitHub Enterprise。

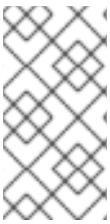
4. 注册的 GitHub OAuth 应用程序的客户端 ID。应用程序必须配置有回调 URL **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**。

5. 对包含 GitHub 发布的客户端 Secret 的 OpenShift Container Platform **Secret** 的引用。

6. 对于 GitHub Enterprise，您必须提供实例的主机名，如 **example.com**。这个值必须与 **/setup/settings** 文件中的 GitHub Enterprise **hostname** 值匹配，且不可包括端口号。如果未设定这个值，则必须定义 **teams** 或 **organizations**。对于 GitHub，请省略此参数。

7. 组织列表。必须设置 **organizations** 或 **teams** 字段，除非设置 **hostname** 字段，或者将 **mappingMethod** 设为 **lookup**。不可与 **teams** 字段结合使用。

8. 团队列表。必须设置 **teams** 或 **organizations** 字段，除非设置 **hostname** 字段，或者将 **mappingMethod** 设为 **lookup**。不可与 **organizations** 字段结合使用。



注意

如果指定了 **organizations** 或 **teams**，只有至少是一个所列组织成员的 GitHub 用户才能登录。如果在 **clientID** 中配置的 GitHub OAuth 应用程序不归该组织所有，则组织所有者必须授予第三方访问权限才能使用此选项。这可以在组织管理员第一次登录 GitHub 时完成，也可以在 GitHub 组织设置中完成。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.6.7. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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

注意

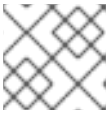
如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 从 OAuth 服务器获取令牌。
只要 **kubeadmin** 用户已被删除，**oc login** 命令就会提供如何访问可以获得令牌的网页的说明。

   您还可以通过使用 web 控制台的 **(?)访问此页面。 Help → Command Line Tools → Copy Login Command**.

3. 登录到集群，提供令牌进行身份验证。

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



注意

这个身份提供程序不支持使用用户名和密码登录。

4. 确认用户登录成功，并显示用户名。

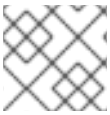
```
$ oc whoami
```

7.7. 配置 GITLAB 身份提供程序

使用 GitLab.com 或任何其他 GitLab 实例作为身份提供程序，配置 **gitlab** 身份提供程序。

7.7.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。



注意

OpenShift Container Platform 用户名不能包括 /、: 和 %。

7.7.2. 关于 GitLab 身份验证

配置 GitLab 身份验证后，用户可以使用其 GitLab 凭证登录 OpenShift Container Platform。

如果使用 GitLab 版本 7.7.0 到 11.0，您可以使用 OAuth 集成进行连接。如果使用 GitLab 版本 11.1 或更高版本，您可以使用 OpenID Connect (OIDC) 进行连接，而不使用 OAuth。

7.7.3. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令创建一个包含字符串的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

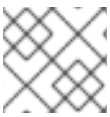
```
apiVersion: v1
kind: Secret
metadata:
  name: <secret_name>
  namespace: openshift-config
type: Opaque
data:
  clientSecret: <base64_encoded_client_secret>
```

- 您可以使用以下命令定义一个包含文件内容的 **Secret** 对象，如证书文件：

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

7.7.4. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。



注意

只有 GitHub Enterprise 需要此步骤。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.7.5. GitLab CR 示例

以下自定义资源 (CR) 显示 GitLab 身份提供程序的参数和可接受值。

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. 此提供程序名称作为前缀放在 GitLab 数字用户 ID 前，以此组成身份名称。它还可用来构建回调 URL。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 注册的 GitLab OAuth 应用程序的客户端 ID 。应用程序必须配置有回调 URL **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**。
4. 对包含 GitLab 发布的客户端 Secret 的 OpenShift Container Platform **Secret** 的引用。

5 GitLab 提供程序的主机 URL。这可以是 **https://gitlab.com/** 或其他自托管 GitLab 实例。

6 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform **ConfigMap** 的引用，以用于验证所配置 URL 的服务器证书。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.7.6. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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

注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 以来自身份提供程序的用户身份登录集群，并在提示时输入密码。

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

3. 确认用户登录成功，并显示用户名。

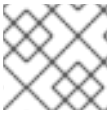
```
$ oc whoami
```

7.8. 配置 GOOGLE 身份提供程序

配置 **google** 身份提供程序，使用 Google 的 OpenID Connect 集成。

7.8.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。



注意

OpenShift Container Platform 用户名不能包括 /、: 和 %。

7.8.2. 关于 Google 身份验证

使用 Google 作为身份提供程序时，任何 Google 用户都能与您的服务器进行身份验证。您可以使用 **hostedDomain** 配置属性，将身份验证限制为特定托管域的成员。

注意

使用 Google 作为身份提供程序要求用户使用 **<namespace_route>/oauth/token/request** 来获取令牌，以便用于命令行工具。

7.8.3. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令创建一个包含字符串的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

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

- 您可以使用以下命令定义一个包含文件内容的 **Secret** 对象，如证书文件：

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

7.8.4. Google CR 示例

以下自定义资源 (CR) 显示 Google 身份提供程序的参数和可接受值。

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. 此提供程序名称作为前缀放在 Google 数字用户 ID 前，以此组成身份名称。它还可用来构建的重定向 URL。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 注册的 Google 项目的客户端 ID 。项目必须配置有重定向 URI **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>**。
4. 对包含 Google 发布的客户端 Secret 的 OpenShift Container Platform **Secret** 的引用。
5. 用于限制登录帐户的托管域。如果使用了 **lookup mappingMethod**，则可选。如果为空，任何 Google 帐户都可进行身份验证。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.8.5. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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

注意

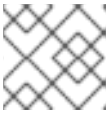
如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 从 OAuth 服务器获取令牌。
只要 **kubeadmin** 用户已被删除，**oc login** 命令就会提供如何访问可以获得令牌的网页的说明。

   您还可以通过使用 web 控制台的 **(?)访问此页面。 Help → Command Line Tools → Copy Login Command**.

3. 登录到集群，提供令牌进行身份验证。

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



注意

这个身份提供程序不支持使用用户名和密码登录。

4. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.9. 配置 OPENID CONNECT 身份提供程序

配置 **oidc** 身份提供程序，使用授权代码流与 OpenID Connect 身份提供程序集成。

7.9.1. 关于 OpenShift Container Platform 中的身份提供程序

默认情况下，集群中只有 **kubeadmin** 用户。要指定身份提供程序，您必须创建一个自定义资源（CR）来描述该身份提供程序并把它添加到集群中。

注意

OpenShift Container Platform 用户名不能包括 /、**:** 和 **%**。

7.9.2. 创建 secret

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **Secret** 对象来包含客户端 secret、客户端证书和密钥。

流程

- 使用以下命令创建一个包含字符串的 **Secret** 对象：

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

提示

您还可以应用以下 YAML 来创建 secret：

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

- 您可以使用以下命令定义一个包含文件内容的 **Secret** 对象，如证书文件：

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

7.9.3. 创建配置映射

身份提供程序使用 **openshift-config** 命名空间中的 OpenShift Container Platform **ConfigMap** 对象来包含证书颁发机构捆绑包。主要用于包含身份提供程序所需的证书捆绑包。

注意

只有 GitHub Enterprise 需要此步骤。

流程

- 使用以下命令，定义包含证书颁发机构的 OpenShift Container Platform **ConfigMap**。证书颁发机构必须存储在 **ConfigMap** 对象的 **ca.crt** 键中。

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

提示

您还可以应用以下 YAML 来创建配置映射：

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

7.9.4. OpenID Connect CR 示例

以下自定义资源 (CR) 显示 OpenID Connect 身份提供程序的参数和可接受值。

如果您必须指定自定义证书捆绑包、额外范围、额外授权请求参数或 **userInfo** URL，请使用完整的 OpenID Connect CR。

标准 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. 此提供程序名称作为前缀放在身份声明值前，以此组成身份名称。它还可用来构建的重定向 URL。
2. 控制如何在此提供程序的身份和 **User** 对象之间建立映射。
3. 在 OpenID 提供程序中注册的客户端的客户端 ID。该客户端必须能够重定向到 **https://oauth-openshift.apps.<cluster_name>.<cluster_domain>/oauth2callback/<idp_provider_name>**。
4. 对包含客户端 secret 的 OpenShift Container Platform **Secret** 对象的引用。
5. 用作身份的声明的列表。使用第一个非空声明。至少需要一个声明。如果列出的声明都没有值，身份验证会失败。例如，这使用返回的 **id_token** 中的 **sub** 声明的值作为用户的身份。
6. OpenID 规范中描述的签发者标识符。必须使用 **https**，且不带查询或分段组件。

完整 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. 可选：对包含 PEM 编码证书颁发机构捆绑包的 OpenShift Container Platform 配置映射的引用，以用于验证所配置 URL 的服务器证书。
2. 除 **openid** 范围外的可选请求范围列表，在授权令牌请求期间使用。
3. 添加至授权令牌请求的附加参数的可选映射。
4. 为此身份置备用户时用作首选用户名的声明的列表。使用第一个非空声明。
5. 用作显示名称的声明列表。使用第一个非空声明。
6. 用作电子邮件地址的声明列表。使用第一个非空声明。

其他资源

- 如需了解所有身份提供程序通用的参数（如 **mappingMethod** ）的信息，请参阅身份提供程序参数。

7.9.5. 在集群中添加身份提供程序

安装集群之后，请在其中添加一个身份提供程序，以便您的用户可以进行身份验证。

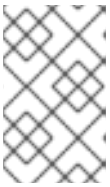
先决条件

- 创建 OpenShift Container Platform 集群。
- 为身份提供程序创建自定义资源（CR）。
- 必须已经以管理员身份登录。

流程

1. 应用定义的 CR：

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



注意

如果一个 CR 不存在，**oc apply** 会创建一个新的 CR，并可能会触发以下警告 **Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**。在这种情况下，您可以忽略这个警告。

2. 从 OAuth 服务器获取令牌。
只要 **kubeadmin** 用户已被删除，**oc login** 命令就会提供如何访问可以获得令牌的网页的说明。

   您还可以通过使用 web 控制台的 **(?)访问此页面。 Help → Command Line Tools→ Copy Login Command**.

3. 登录到集群，提供令牌进行身份验证。

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

注意

如果您的 OpenID Connect 身份提供程序支持资源所有者密码凭证（ROPC）授权流，您可以使用用户名和密码登录。您可能需要执行相应的步骤，为身份提供程序启用 ROPC 授权流。

在 OpenShift Container Platform 中配置 OIDC 身份提供程序后，您可以使用以下命令登录，以提示您的用户名和密码：

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

4. 确认用户登录成功，并显示用户名。

```
$ oc whoami
```

7.9.6. 使用 web 控制台配置身份提供程序

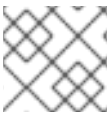
通过 web 控制台而非 CLI 配置身份提供程序 (IDP)。

先决条件

- 您必须以集群管理员身份登录到 web 控制台。

流程

1. 导航至 **Administration → Cluster Settings**。
2. 在 **Global Configuration** 选项卡下，点 **OAuth**。
3. 在 **Identity Providers** 部分中，从 **Add** 下拉菜单中选择您的身份提供程序。



注意

您可以通过 web 控制台来指定多个 IDP，而不会覆盖现有的 IDP。

第 8 章 使用 RBAC 定义和应用权限

8.1. RBAC 概述

基于角色的访问控制 (RBAC) 对象决定是否允许用户在项目内执行给定的操作。

集群管理员可以使用集群角色和绑定来控制谁对 OpenShift Container Platform 平台本身和所有项目具有各种访问权限等级。

开发人员可以使用本地角色和绑定来控制谁有权访问他们的项目。请注意，授权是与身份验证分开的一个步骤，身份验证更在于确定执行操作的人的身份。

授权通过使用以下几项来管理：

授权对象	描述
规则	一组对象上允许的操作集合。例如，用户或服务帐户能否**创建（create）** Pod。
角色	规则的集合。可以将用户和组关联或绑定到多个角色。
绑定	用户和/组与角色之间的关联。

控制授权的 RBAC 角色和绑定有两个级别：

RBAC 级别	描述
集群 RBAC	对所有项目均适用的角色和绑定。*集群角色*存在于集群范围，*集群角色绑定*只能引用集群角色。
本地 RBAC	作用于特定项目的角色和绑定。虽然*本地角色*只存在于单个项目中，但本地角色绑定可以*同时*引用集群和本地角色。

集群角色绑定是存在于集群级别的绑定。角色绑定存在于项目级别。集群角色 *view* 必须使用本地角色绑定来绑定到用户，以便该用户能够查看项目。只有集群角色不提供特定情形所需的权限集合时才应创建本地角色。

这种双级分级结构允许通过集群角色在多个项目间重复使用，同时允许通过本地角色在个别项目中自定义。

在评估过程中，同时使用集群角色绑定和本地角色绑定。例如：

1. 选中集群范围的“allow”规则。
2. 选中本地绑定的“allow”规则。
3. 默认为拒绝。

8.1.1. 默认集群角色

OpenShift Container Platform 包括了一组默认的集群角色，您可以在集群范围或本地将它们绑定到用户和组。

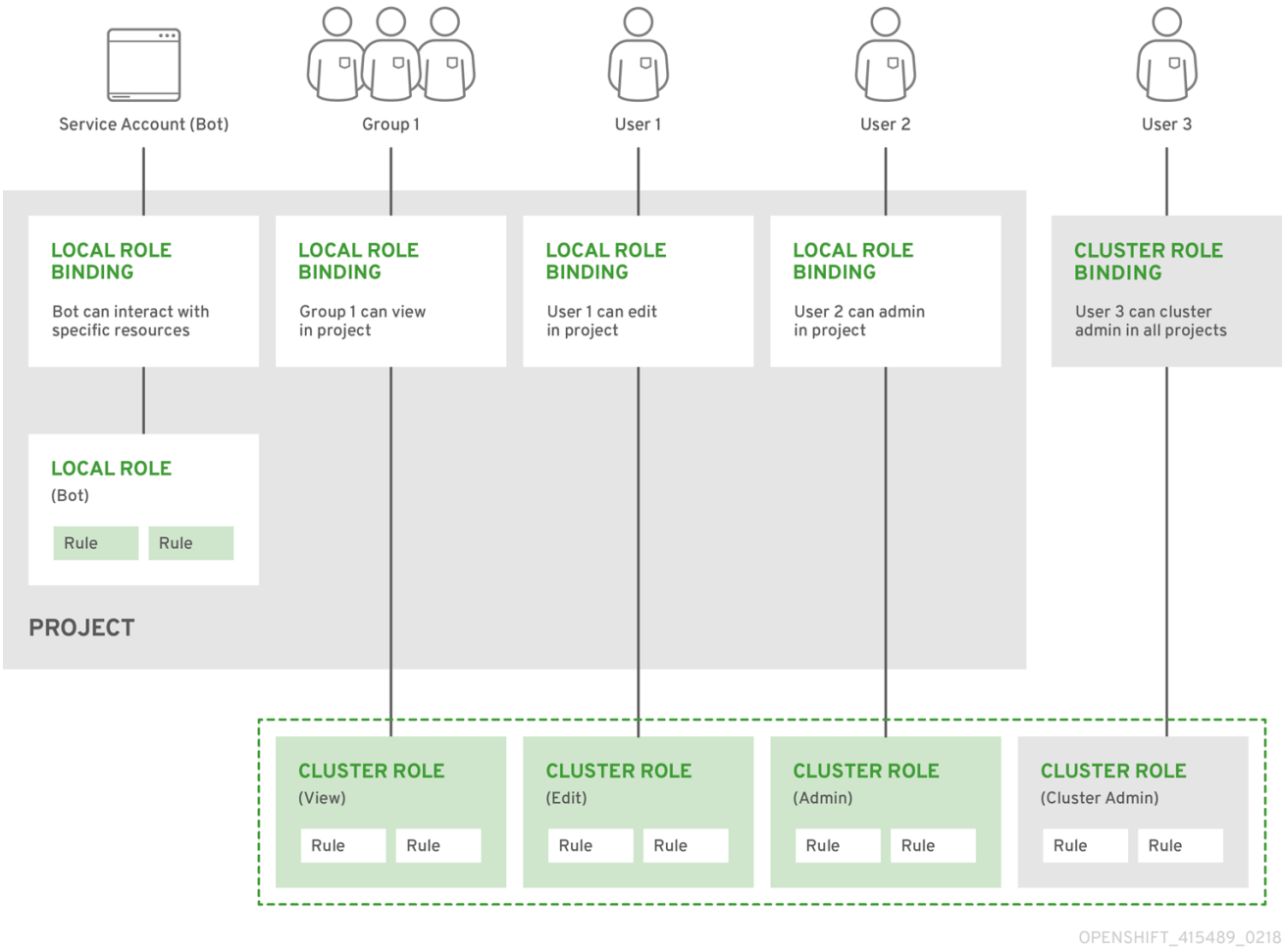
重要

不建议手动修改默认集群角色。对这些系统角色的修改可能会阻止集群正常工作。

默认集群角色	描述
admin	项目管理者。如果在本地绑定中使用，则 **admin** 有权查看项目中的任何资源，并且修改项目中除配额外的任何资源。
basic-user	此用户可以获取有关项目和用户的基本信息。
cluster-admin	此超级用户可以在任意项目中执行任何操作。当使用本地绑定来绑定一个用户时，这些用户可以完全控制项目中每一资源的配额和所有操作。
cluster-status	此用户可以获取基本的集群状态信息。
cluster-reader	用户可以获取或查看大多数对象，但不能修改它们。
edit	此用户可以修改项目中大多数对象，但无权查看或修改角色或绑定。
self-provisioner	此用户可以创建自己的项目。
view	此用户无法进行任何修改，但可以查看项目中的大多数对象。不能查看或修改角色或绑定。

请注意本地和集群绑定之间的区别。例如，如果使用本地角色绑定将 **cluster-admin** 角色绑定到一个用户，这可能看似该用户具有了集群管理员的特权。事实并非如此。将 **cluster-admin** 绑定到项目里的某一用户，仅会将该项目的超级管理员特权授予这一用户。该用户具有集群角色 **admin** 的权限，以及该项目的一些额外权限，例如能够编辑项目的速率限制。通过 web 控制台 UI 操作时此绑定可能会令人混淆，因为它不会列出绑定到真正集群管理员的集群角色绑定。然而，它会列出可用来本地绑定 **cluster-admin** 的本地角色绑定。

下方展示了集群角色、本地角色、集群角色绑定、本地角色绑定、用户、组和服务帐户之间的关系。



8.1.2. 评估授权

OpenShift Container Platform 使用以下几项来评估授权：

身份

用户名以及用户所属组的列表。

操作

您执行的操作。在大多数情况下，这由以下几项组成：

- **项目**：您访问的项目。项目是一种附带额外注解的 Kubernetes 命名空间，使一个社区的用户可以在与其他社区隔离的前提下组织和管理其内容。
- **操作动词**：操作本身：**get**、**list**、**create**、**update**、**delete**、**deletecollection** 或 **watch**。
- **资源名称**：您访问的 API 端点。

绑定

绑定的完整列表，用户或组与角色之间的关联。

OpenShift Container Platform 通过以下几个步骤评估授权：

1. 使用身份和项目范围操作来查找应用到用户或所属组的所有绑定。
2. 使用绑定来查找应用的所有角色。
3. 使用角色来查找应用的所有规则。

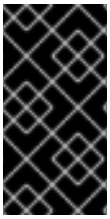
4. 针对每一规则检查操作，以查找匹配项。

5. 如果未找到匹配的规则，则默认拒绝该操作。

提示

请记住，用户和组可以同时关联或绑定到多个角色。

项目管理员可以使用 CLI 查看本地角色和绑定信息，包括与每个角色关联的操作动词和资源的一览表。



重要

通过本地绑定来绑定到项目管理员的集群角色会限制在一个项目内。不会像授权给 **cluster-admin** 或 **system:admin** 的集群角色那样在集群范围绑定。

集群角色是在集群级别定义的角色，但可在集群级别或项目级别进行绑定。

8.1.2.1. 集群角色聚合

默认的 admin、edit、view 和 cluster-reader 集群角色支持集群角色聚合，其中每个角色的集群规则可在创建了新规则时动态更新。只有通过创建自定义资源扩展 Kubernetes API 时，此功能才有意义。

8.2. 项目和命名空间

Kubernetes *命名空间*提供设定集群中资源范围的机制。Kubernetes 文档中提供有关命名空间的更多信息。

命名空间为以下对象提供唯一范围：

- 指定名称的资源，以避免基本命名冲突。
- 委派给可信用户的管理授权。
- 限制社区资源消耗的能力。

系统中的大多数对象都通过命名空间来设定范围，但一些对象不在此列且没有命名空间，如节点和用户。

*项目*是附带额外注解的 Kubernetes 命名空间，是管理常规用户资源访问权限的集中载体。通过项目，一个社区的用户可以在与其他社区隔离的前提下组织和管理其内容。用户必须由管理员授予对项目的访问权限；或者，如果用户有权创建项目，则自动具有自己创建项目的访问权限。

项目可以有单独的 **name**、**displayName** 和 **description**。

- 其中必备的 **name** 是项目的唯一标识符，在使用 CLI 工具或 API 时最常见。名称长度最多为 63 个字符。
- 可选的 **displayName** 是项目在 web 控制台中的显示形式（默认为 **name**）。
- 可选的 **description** 可以为项目提供更加详细的描述，也可显示在 web 控制台中。

每个项目限制了自己的一组：

对象	描述
对象（object）	Pod、服务和复制控制器等。
策略（policy）	用户能否对对象执行操作的规则。
约束（constraint）	对各种对象进行限制的配额。
服务帐户	服务帐户自动使用项目中对象的指定访问权限进行操作。

集群管理员可以创建项目，并可将项目的管理权限委派给用户社区的任何成员。集群管理员也可以允许开发人员创建自己的项目。

开发人员和管理员可以使用 CLI 或 Web 控制台与项目交互。

8.3. 默认项目

OpenShift Container Platform 附带若干默认项目，名称以 **openshift--** 开头的项目对用户而言最为重要。这些项目托管作为 Pod 运行的主要组件和其他基础架构组件。在这些命名空间中创建的带有关键 (critical) Pod 注解的 Pod 是很重要的，它们可以保证被 kubelet 准入。在这些命名空间中为主要组件创建的 Pod 已标记为“critical”。

注意

您无法将 SCC 分配给在以下某一默认命名空间中创建的 Pod: **default**、**kube-system**、**kube-public**、**openshift-node**、**openshift-infra**、**openshift**。您不能使用这些命名空间用来运行 pod 或服务。

8.4. 查看集群角色和绑定

通过 **oc describe** 命令，可以使用 **oc** CLI 来查看集群角色和绑定。

先决条件

- 安装 **oc** CLI。
- 获取查看集群角色和绑定的权限。

在集群范围内绑定了 **cluster-admin** 默认集群角色的用户可以对任何资源执行任何操作，包括查看集群角色和绑定。

流程

1. 查看集群角色及其关联的规则集：

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

输**出示例**

```
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. 查看当前的集群角色绑定集合，这显示绑定到不同角色的用户和组：

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

输**出示例**

```
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. 查看本地角色和绑定

使用 **oc describe** 命令通过 **oc** CLI 来查看本地角色和绑定。

先决条件

- 安装 **oc** CLI。
- 获取查看本地角色和绑定的权限：
  - 在集群范围内绑定了 **cluster-admin** 默认集群角色的用户可以对任何资源执行任何操作，包括查看本地角色和绑定。
  - 本地绑定了 **admin** 默认集群角色的用户可以查看并管理项目中的角色和绑定。

流程

1. 查看当前本地角色绑定集合，这显示绑定到当前项目的不同角色的用户和组：

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

2. 要查其他项目的本地角色绑定，请向命令中添加 **-n** 标志：

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

输**出示例**

```
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. 向用户添加角色

可以使用 **oc adm** 管理员 CLI 管理角色和绑定。

将角色绑定或添加到用户或组可让用户或组具有该角色授予的访问权限。您可以使用 **oc adm policy** 命令向用户和组添加和移除角色。

您可以将任何默认集群角色绑定到项目中的本地用户或组。

流程

1. 向指定项目中的用户添加角色：

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

例如，您可以运行以下命令，将 **admin** 角色添加到 **joe** 项目中的 **alice** 用户：

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

提示

您还可以应用以下 YAML 向用户添加角色：

```
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. 查看本地角色绑定，并在输出中验证添加情况：

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

例如，查看 **joe** 项目的本地角色绑定：

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

输**出示例**

```
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** 用户已添加到 **admins RoleBinding**。

8.7. 创建本地角色

您可以为项目创建本地角色，然后将其绑定到用户。

流程

1. 要为项目创建本地角色，请运行以下命令：

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

在此命令中，指定：

- **<name>**，本地角色的名称
- **<verb>**，以逗号分隔的、应用到角色的操作动词列表
- **<resource>**，角色应用到的资源
- **<project>**，项目名称

例如，要创建一个本地角色来允许用户查看 **blue** 项目中的 Pod，请运行以下命令：

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

2. 要将新角色绑定到用户，运行以下命令：

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

8.8. 创建集群角色

您可以创建集群角色。

流程

1. 要创建集群角色，请运行以下命令：

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

在此命令中，指定：

- **<name>**，本地角色的名称
- **<verb>**，以逗号分隔的、应用到角色的操作动词列表
- **<resource>**，角色应用到的资源

例如，要创建一个集群角色来允许用户查看 Pod，请运行以下命令：

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

8.9. 本地角色绑定命令

在使用以下操作为本地角色绑定管理用户或组的关联角色时，可以使用 **-n** 标志来指定项目。如果未指定，则使用当前项目。

您可以使用以下命令进行本地 RBAC 管理。

表 8.1. 本地角色绑定操作

命令	描述
$ oc adm policy who-can *<verb> <resource>*	指出哪些用户可以对某一资源执行某种操作。
$ oc adm policy add-role-to-user *<role> <username>*	将指定角色绑定到当前项目中的指定用户。
$ oc adm policy remove-role-from-user *<role> <username>*	从当前项目中的指定用户移除指定角色。
$ oc adm policy remove-user *<username>*	移除当前项目中的指定用户及其所有角色。
$ oc adm policy add-role-to-group *<role> <groupname>*	将给定角色绑定到当前项目中的指定组。
$ oc adm policy remove-role-from-group *<role> <groupname>*	从当前项目中的指定组移除给定角色。
$ oc adm policy remove-group *<groupname>*	移除当前项目中的指定组及其所有角色。

8.10. 集群角色绑定命令

您也可以使用以下操作管理集群角色绑定。因为集群角色绑定使用没有命名空间的资源，所以这些操作不使用 **-n** 标志。

表 8.2. 集群角色绑定操作

命令	描述
$ oc adm policy add-cluster-role-to-user *<role> <username>*	将给定角色绑定到集群中所有项目的指定用户。
$ oc adm policy remove-cluster-role-from-user *<role> <username>*	从集群中所有项目的指定用户移除给定角色。
$ oc adm policy add-cluster-role-to-group *<role> <groupname>*	将给定角色绑定到集群中所有项目的指定组。
$ oc adm policy remove-cluster-role-from-group *<role> <groupname>*	从集群中所有项目的指定组移除给定角色。

8.11. 创建集群管理员

需要具备 **cluster-admin** 角色才能在 OpenShift Container Platform 集群上执行管理员级别的任务，例如修改集群资源。

先决条件

- 您必须已创建了要定义为集群管理员的用户。

流程

- 将用户定义为集群管理员：

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

第 9 章 移除 KUBEADMIN 用户

9.1. KUBEADMIN 用户

OpenShift Container Platform 在安装过程完成后会创建一个集群管理员 **kubeadmin**。

此用户自动具有 **cluster-admin** 角色，并视为集群的 root 用户。其密码是动态生成的，对 OpenShift Container Platform 环境中是唯一的。安装完成后，安装程序的输出中会包括这个密码。例如：

```
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. 移除 KUBEADMIN 用户

在定义了身份提供程序并创建新的 **cluster-admin** 用户后，您可以移除 **kubeadmin** 来提高集群安全性。

警告

如果在另一用户成为 **cluster-admin** 前按照这个步骤操作，则必须重新安装 OpenShift Container Platform。此命令无法撤销。

先决条件

- 必须至少配置了一个身份提供程序。
- 必须向用户添加了 **cluster-admin** 角色。
- 必须已经以管理员身份登录。

流程

- 移除 **kubeadmin** Secret：

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

第 10 章 了解并创建服务帐户

10.1. 服务帐户概述

服务帐户是一种 OpenShift Container Platform 帐户，它允许组件直接访问 API。服务帐户是各个项目中存在的 API 对象。服务帐户为控制 API 访问提供了灵活的方式，不需要共享常规用户的凭证。

使用 OpenShift Container Platform CLI 或 web 控制台时，您的 API 令牌会为您与 API 进行身份验证。您可以将组件与服务帐户关联，以便组件能够访问 API 且无需使用常规用户的凭证。例如，借助服务帐户：

- 复制控制器可以发出 API 调用来创建或删除 Pod。
- 容器内的应用程序可以发出 API 调用来进行发现。
- 外部应用程序可以发出 API 调用来进行监控或集成。

每个服务帐户的用户名都源自于其项目和名称：

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

每一服务帐户也是以下两个组的成员：

组	描述
system:serviceaccounts	包含系统中的所有服务帐户。
system:serviceaccounts:<project>	包含指定项目中的所有服务帐户。

每个服务帐户自动包含两个 secret：

- API 令牌
- OpenShift Container Registry 的凭证

生成的 API 令牌和 registry 凭证不会过期，但可通过删除 secret 来撤销它们。当删除 secret 时，系统会自动生成一个新 secret 来取代它。

10.2. 创建服务帐户

您可以在项目中创建服务帐户，并通过将其绑定到角色为该帐户授予权限。

流程

1. 可选：查看当前项目中的服务帐户：

```
$ oc get sa
```

输出示例

```
NAME      SECRETS   AGE
```

```
builder   2        2d
default   2        2d
deployer  2        2d
```

2. 在当前项目中创建新服务帐户：

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

1 要在另一项目中创建服务帐户，请指定 **-n <project_name>**。

输出示例

```
serviceaccount "robot" created
```

提示

您还可以应用以下 YAML 来创建服务帐户：

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

3. 可选：查看服务帐户的 secret ：

```
$ oc describe sa robot
```

输出示例

```
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. 为服务帐户授予角色的示例

您可以像为常规用户帐户授予角色一样，为服务帐户授予角色。

- 您可以修改当前项目的服务帐户。例如，将 **view** 角色添加到 **top-secret** 项目中的 **robot** 服务帐户：

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

-

提示

您还可以应用以下 YAML 来添加角色：

```
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
```

- 您也可以向项目中的特定服务帐户授予访问权限。例如，在服务帐户所属的项目中，使用 **-z** 标志并指定 **<service_account_name>**

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

重要

如果要向项目中的特定服务帐户授予访问权限，请使用 **-z** 标志。使用此标志有助于预防拼写错误，并确保只为指定的服务帐户授予访问权限。

提示

您还可以应用以下 YAML 来添加角色：

```
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>
```

- 要修改不同的命名空间，可以使用 **-n** 选项指定它要应用到的项目命名空间，如下例所示。
  - 例如，允许所有项目中的所有服务帐户查看 **my-project** 项目中的资源：

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

提示

您还可以应用以下 YAML 来添加角色：

```
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
```

- 允许 **managers** 项目中的所有服务帐户编辑 **my-project** 项目中的资源：

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

提示

您还可以应用以下 YAML 来添加角色：

```
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
```

第 11 章 在应用程序中使用服务帐户

11.1. 服务帐户概述

服务帐户是一种 OpenShift Container Platform 帐户，它允许组件直接访问 API。服务帐户是各个项目中存在的 API 对象。服务帐户为控制 API 访问提供了灵活的方式，不需要共享常规用户的凭证。

使用 OpenShift Container Platform CLI 或 web 控制台时，您的 API 令牌会为您与 API 进行身份验证。您可以将组件与服务帐户关联，以便组件能够访问 API 且无需使用常规用户的凭证。例如，借助服务帐户：

- 复制控制器可以发出 API 调用来创建或删除 Pod。
- 容器内的应用程序可以发出 API 调用来进行发现。
- 外部应用程序可以发出 API 调用来进行监控或集成。

每个服务帐户的用户名都源自于其项目和名称：

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

每一服务帐户也是以下两个组的成员：

组	描述
system:serviceaccounts	包含系统中的所有服务帐户。
system:serviceaccounts:<project>	包含指定项目中的所有服务帐户。

每个服务帐户自动包含两个 secret：

- API 令牌
- OpenShift Container Registry 的凭证

生成的 API 令牌和 registry 凭证不会过期，但可通过删除 secret 来撤销它们。当删除 secret 时，系统会自动生成一个新 secret 来取代它。

11.2. 默认服务帐户

OpenShift Container Platform 集群包含用于集群管理的默认服务帐户，并且为各个项目生成更多服务帐户。

11.2.1. 默认集群服务帐户

几个基础架构控制器使用服务帐户凭证运行。服务器启动时在 OpenShift Container Platform 基础架构项目 (**openshift-infra**) 中创建以下服务帐户，并授予其如下集群范围角色：

服务帐户	描述
replication-controller	分配 **system:replication-controller** 角色
deployment-controller	分配 **system:deployment-controller** 角色。
build-controller	分配 **system:build-controller** 角色。另外，**build-controller** 服务帐户也包含在特权安全上下文约束中，以创建特权构建 Pod。

11.2.2. 默认项目服务帐户和角色

每个项目中会自动创建三个服务帐户：

服务帐户	使用方法
builder	由构建 Pod 使用。被授予 **system:image-builder** 角色，允许使用内部 Docker registry 将镜像推送到项目中的任何镜像流。
deployer	由部署 Pod 使用并被授予 **system:deployer** 角色，允许查看和修改项目中的复制控制器和 Pod。
default	用来运行其他所有 Pod，除非指定了不同的服务帐户。

项目中的所有服务帐户都会被授予 **system:image-puller** 角色，允许使用内部容器镜像 registry 从项目中的任何镜像流拉取镜像。

11.3. 创建服务帐户

您可以在项目中创建服务帐户，并通过将其绑定到角色为该帐户授予权限。

流程

1. 可选：查看当前项目中的服务帐户：

```
$ oc get sa
```

输出示例

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

2. 在当前项目中创建新服务帐户：

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

1 要在另一项目中创建服务帐户，请指定 **-n <project_name>**。

输出示例

```
serviceaccount "robot" created
```

提示

您还可以应用以下 YAML 来创建服务帐户：

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

3. 可选：查看服务帐户的 secret ：

```
$ oc describe sa robot
```

输出示例

```
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. 在外部使用服务帐户凭证

您可以将服务帐户的令牌分发给必须通过 API 身份验证的外部应用程序。

若要拉取镜像，经过身份验证的用户必须具有所请求的 **imagestreams/layers** 的 **get** 权限。要推送镜像，经过身份验证的用户必须具有所请求的 **imagestreams/layers** 的 **update** 权限。

默认情况下，一个项目中的所有服务帐户都有权拉取同一项目中的任何镜像，而 **builder** 服务帐户则有权在同一项目中推送任何镜像。

流程

1. 查看服务帐户的 API 令牌：

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

例如：

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

输出示例

```
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. 使用您获取的令牌进行登录：

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

输出示例

```
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. 确认您已经以服务帐户登录：

```
$ oc whoami
```

输出示例

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

第 12 章 使用服务帐户作为 OAUTH 客户端

12.1. 服务帐户作为 OAUTH 客户端

您可以使用服务帐户，作为受约束形式的 OAuth 客户端。服务帐户只能请求范围的子集，允许访问服务帐户本身的命名空间中的一些基本用户信息和基于角色的功能：

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

在将服务帐户用作 OAuth 客户端时：

- **client_id** 是 **system:serviceaccount:<service_account_namespace>:<service_account_name>**。
- **client_secret** 可以是该服务帐户的任何 API 令牌。例如：

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

- 要获取 **WWW-Authenticate** 质询，请将服务帐户上的 **serviceaccounts.openshift.io/oauth-want-challenges** 注解设置为 **true**。
- **redirect_uri** 必须与服务帐户上的注解匹配。

12.1.1. 重定向作为 OAuth 客户端的服务帐户的 URI

注解键必须具有前缀 **serviceaccounts.openshift.io/oauth-redirecturi.** 或 **serviceaccounts.openshift.io/oauth-redirectreference.**，例如：

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

采用最简单形式时，注解可用于直接指定有效的重定向 URI。例如：

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

上例中的 **first** 和 **second** 后缀用于分隔两个有效的重定向 URI。

在更复杂的配置中，静态重定向 URI 可能还不够。例如，您可能想要路由的所有入口都被认为是有效的。这时可使用通过 **serviceaccounts.openshift.io/oauth-redirectreference.** 前缀的动态重定向 URI。

例如：

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

由于此注解的值包含序列化 JSON 数据，因此在扩展格式中可以更轻松地查看：

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

您现在可以看到，**OAuthRedirectReference** 允许引用名为 **jenkins** 的路由。因此，该路由的所有入口现在都被视为有效。**OAuthRedirectReference** 的完整规格是：

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

1. **kind** 指的是被引用对象的类型。目前，只支持 **route**。
2. **name** 指的是项目的名称。对象必须与服务帐户位于同一命名空间中。
3. **group** 指的是对象所属的组。此项留空，因为路由的组是空字符串。

可以合并这两个注解前缀，来覆盖引用对象提供的数据。例如：

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

first 后缀用于将注解绑在一起。假设 **jenkins** 路由曾具有入口 **https://example.com**，现在**https://example.com/custompath** 被视为有效，但 **https://example.com** 视为无效。部分提供覆盖数据的格式如下：

类型	语法
Scheme	"https://"
主机名	"//website.com"
端口	"//:8000"
路径	"examplepath"

注意

指定主机名覆盖将替换被引用对象的主机名数据，这不一定是需要的行为。

以上语法的任何组合都可以使用以下格式进行合并：

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

同一对象可以被多次引用，以获得更大的灵活性：

```
"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\"}}"
```

假设名为 **jenkins** 的路由具有入口 **https://example.com**，则 **https://example.com:8000** 和 **https://example.com/custompath** 都被视为有效。

可以同时使用静态和动态注解，以实现所需的行为：

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

第 13 章 界定令牌作用域

13.1. 关于界定令牌作用域

您可以创建有作用域令牌，将某些权限委派给其他用户或服务帐户。例如，项目管理员可能希望委派创建 Pod 的权限。

有范围的令牌用来标识给定用户，但仅限于其范围指定的特定操作。只有具有 **cluster-admin** 角色的用户才能创建有范围的令牌。

通过将令牌的范围集合转换为 **PolicyRule** 集合来评估其范围。然后，请求会与这些规则进行匹配。请求属性必须至少匹配其中一条范围规则，才能传递给 "normal" 授权程序进行进一步授权检查。

13.1.1. 用户范围

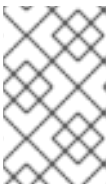
用户范围主要用于获取给定用户的信息。它们是基于意图的，因此会自动为您创建规则：

- **user:full** - 允许使用用户的所有权限对 API 进行完全的读/写访问。
- **user:info** - 允许只读访问用户的信息，如名称和组。
- **user:check-access** - 允许访问 **self-localsubjectaccessreviews** 和 **self-subjectaccessreviews**。这些是在请求对象中传递空用户和组的变量。
- **user:list-projects** - 允许只读访问，可以列出用户可访问的项目。

13.1.2. 角色范围

角色范围允许您具有与给定角色相同等级的访问权限，该角色通过命名空间过滤。

- **role:<cluster-role name>:<namespace or * for all>** - 将范围限定为集群角色指定的规则，但仅在指定的命名空间中。



注意

注意：这可防止升级访问权限。即使角色允许访问 secret、角色绑定和角色等资源，但此范围会拒绝访问这些资源。这有助于防止意外升级。许多人认为 **edit** 等角色并不是升级角色，但对于访问 secret 而言，这的确是升级角色。

- **role:<cluster-role name>:<namespace or * for all>:!** - 这与上例相似，但因为包含感叹号而使得此范围允许升级访问权限。

第 14 章 使用绑定的服务帐户令牌

您可以使用绑定服务帐户令牌，它可以提高与云供应商身份访问管理 (IAM) 服务（如 AWS IAM)集成的能力。

14.1. 关于绑定服务帐户令牌

您可以使用绑定服务帐户令牌来限制给定服务帐户令牌的权限范围。这些令牌受使用者和时间的限制。这有助于服务帐户到 IAM 角色的身份验证以及挂载到 pod 的临时凭证的生成。您可以使用卷投射和 TokenRequest API 来请求绑定的服务帐户令牌。

14.2. 使用卷投射配置绑定服务帐户令牌

您可以使用卷投射，将 pod 配置为请求绑定的服务帐户令牌。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已创建了一个服务帐户。这里假定服务帐户命名为 **build-robot**。

流程

1. 可选：设置服务帐户签发者。
   如果绑定令牌仅在集群中使用，则通常不需要这一步。

警告

如果您更新 **serviceAccountIssuer** 字段，且已使用绑定令牌，则具有之前签发者值的所有绑定令牌都将无效。除非绑定令牌的拥有者明确支持签发者更改，否则该拥有者不会请求一个新的绑定令牌，直到 pod 重启为止。

您可以通过手动重启集群中的所有 pod 或执行滚动节点重启来强制所有拥有者请求新的绑定令牌。在执行任一操作前，等待 Kubernetes API 服务器 pod 的新修订版本来向您的服务帐户签发者更改。

   a. 编辑 **cluster Authentication** 对象：

```
$ oc edit authentications cluster
```

   b. 将 **spec.serviceAccountIssuer** 字段设置为所需的服务帐户签发者：

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

1 这个值应该是 URL，通过这个 URL，绑定令牌的接收方可以从中查找验证令牌签名所需的公钥。默认为 **https://kubernetes.default.svc**。

c. 保存文件以使改变生效。

d. 等待 Kubernetes API 服务器 pod 的新修订版本推出。所有节点更新至新修订版本可能需要几分钟时间。运行以下命令:

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

查看 Kubernetes API 服务器的 **NodeInstallerProgressing** 状态条件，以验证所有节点是否处于最新的修订版本。在更新成功后，输出会显示 **AllNodesAtLatestRevision**：

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

(1) 在本例中，最新的修订版本号为 **12**。

如果输出显示的信息类似于以下消息之一，则更新仍在进行中。等待几分钟后重试。

- **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. 可选：强制拥有者通过执行滚动节点重启或重启集群中的所有 pod 来请求新的绑定令牌。

- 执行滚动节点重启：

> **警告**
>
> 如果您已在集群上运行自定义工作负载，则不建议执行滚动节点重启，因为它可能会导致服务中断。相反，手动重启集群中的所有 pod。

按顺序重启节点。等待节点完全可用，然后重启下一个节点。有关如何排空、重启并将节点标记为可以调度的说明，请参阅*正常重启节点*。

- 手动重启集群中的所有 pod:

> **警告**
>
> 此命令会导致服务中断，因为它将删除每个命名空间中运行的所有 pod。这些 Pod 会在删除后自动重启。

运行以下命令:

```
$ 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. 使用卷投影将 pod 配置为使用绑定服务帐户令牌。

   a. 创建名为 **pod-projected-svc-token.yaml** 的文件，其内容如下：

```
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) 对现有服务帐户的引用。

   (2) 相对于文件挂载点的相对路径，用于将令牌放入。

   (3) （可选）设置服务帐户令牌的到期时间（以秒为单位）。默认为 3600 秒（1 小时），且必须至少为 600 秒（10 分钟）。如果令牌使用的时间超过这个值的 80%，或者超过 24 小时，则 kubelet 会开始尝试轮转令牌。

   (4) （可选）设置令牌的预期使用者。令牌的接收者应验证接收者身份是否与令牌的使用声明匹配，否则应拒绝令牌。使用者默认为 API 服务器的标识符。

   b. 创建 pod：

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

   Kubelet 代表 pod 请求并存储令牌，使 pod 可以在一个可配置的文件路径中获得令牌，并在该令牌接近到期时刷新令牌。

3. 使用绑定令牌的应用程序需要在令牌轮转时重新载入令牌。
   如果令牌使用的时间超过这个值的 80%，或者超过 24 小时，则 kubelet 会轮转令牌。

其他资源

- 正常重新引导节点

第 15 章 管理安全性上下文约束

15.1. 关于安全性上下文约束

与 RBAC 资源控制用户访问的方式类似，管理员可以使用*安全性上下文约束* (SCC) 来控制 Pod 的权限。这些权限包括 Pod 可以执行的操作以及它们可以访问的资源。您可以使用 SCC 定义 Pod 运行必须满足的一组条件，以便其能被系统接受。

通过安全性上下文约束，管理员可以控制：

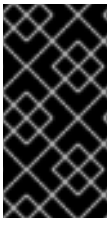
- pod 是否可以使用 **allowPrivilegedContainer** 标志运行特权容器。
- 使用 **allowPrivilegeEscalation** 标记限制 pod。
- 容器可以请求的功能
- 将主机目录用作卷
- 容器的 SELinux 上下文
- 容器用户 ID
- 使用主机命名空间和网络
- 拥有 pod 卷的 **FSGroup** 的分配
- 允许的补充组的配置
- 容器是否需要对其 root 文件系统进行写访问权限
- 卷类型的使用
- 允许的 **seccomp** 配置集的配置

重要

不要在 OpenShift Container Platform 中的任何命名空间上设置 **openshift.io/run-level** 标签。此标签供内部 OpenShift Container Platform 组件用来管理主要 API 组的启动，如 Kubernetes API 服务器和 OpenShift API 服务器。如果设置了 **openshift.io/run-level** 标签，则不会将 SCC 应用到该命名空间中的 pod，从而导致该命名空间中运行的任何工作负载都具有高度特权。

15.1.1. 默认安全性上下文约束

集群包含多个默认安全性上下文约束 (SCC)，如下表所述。将 Operator 或其他组件安装到 OpenShift Container Platform 时，可能会安装额外的 SCC。



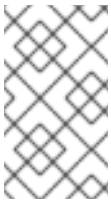
重要

不要修改默认 SCC。自定义默认 SCC 可能会导致在一些平台 Pod 部署或 OpenShift Container Platform 升级时出现问题。在 OpenShift Container Platform 某些版本之间的升级过程中，默认 SCC 的值被重置为默认值，这会丢弃对这些 SCC 的所有自定义。相反，请根据需要创建新的 SCC。

表 15.1. 默认安全性上下文约束

安全性上下文约束	描述
anyuid	提供 **restricted** SCC 的所有功能，但允许用户使用任何 UID 和任何 GID 运行。
hostaccess	允许访问所有主机命名空间，但仍要求使用分配至命名空间的 UID 和 SELinux 上下文运行容器集。 **警告** 此 SCC 允许主机访问命名空间、文件系统和 PID。它应当仅由受信任的容器集使用。请谨慎授予。
hostmount-anyuid	提供 **restricted** SCC 的所有功能，但允许主机以任何 UID 和系统中的任何 GID 挂载和运行。 **警告** 此 SCC 允许主机文件系统作为任何 UID 访问，包括 UID 0。请谨慎授予。
hostnetwork	允许使用主机网络和主机端口，但仍要求使用分配至命名空间的 UID 和 SELinux 上下文运行容器集。 **警告** 如果在 control plane 主机上运行额外的工作负载（也称为 master 主机），在提供 **hostnetwork** 访问权限时要谨慎。在 control plane 主机上运行 **hostnetwork** 的工作负载是集群中的 root 用户，必须相应地信任它。

安全性上下文约束	描述
node-exporter	用于 Prometheus 节点导出器。 **警告** 此 SCC 允许主机文件系统作为任何 UID 访问，包括 UID 0。请谨慎授予。
nonroot	提供 **restricted** SCC 的所有功能，但允许用户使用任何非 root UID 运行。用户必须指定 UID，否则必须在容器运行时清单中指定。
privileged	允许访问所有特权和主机功能，并且能够以任何用户、任何组、任何 FSGroup 以及任何 SELinux 上下文运行。 **警告** 这是最宽松的 SCC，应仅用于集群管理。请谨慎授予。 **特权** SCC 允许： • 用户运行特权 Pod • Pod 将主机目录挂载为卷 • Pod 以任意用户身份运行 • Pod 使用任意 MCS 标签运行 • Pod 使用主机的 IPC 命名空间 • Pod 使用主机的 PID 命名空间 • Pod 使用任何 FSGroup • Pod 使用任何补充组 • Pod 使用任何 seccomp 配置集 • Pod 请求任何功能 **注意** 在 Pod 规格中设置 **privileged: true** 并不一定会选择**特权** SCC。如果用户有权使用，则具有 **allowPrivilegedContainer: true** 并有最高优先级的 SCC 会被选择。

安全性上下文约束	描述
restricted	拒绝访问所有主机功能，并且要求使用 UID 运行容器集，以及分配至命名空间的 SELinux 上下文。这是一个新安装可以提供的最严格的 SCC，经过身份验证的用户会默认使用它。 **受限的** SCC： ● 确保 Pod 无法以特权方式运行 ● 确保 Pod 无法挂载主机目录卷 ● 要求 Pod 以预先分配的 UID 范围内的用户运行 ● 要求 pod 使用预先分配的 MCS 标签运行 ● 允许 Pod 使用任何 FSGroup ● 允许 pod 使用任何补充组  **注意** 有限制的 SCC 是系统默认提供的、有最严格限制的 SCC。但是，您可以创建一个更加严格的自定义 SCC。例如，您可以创建一个 SCC，它将 **readOnlyRootFS** 限制为 **true**，并允许 **PrivilegeEscalation** 为 **false**。

15.1.2. 安全性上下文约束设置

安全性上下文约束 (SCC) 由控制 Pod 可访问的安全功能的设置和策略组成。这些设置分为三个类别：

类别	描述
由布尔值控制	此类型的字段默认为限制性最强的值。例如，**AllowPrivilegedContainer** 若未指定，则始终设为 **false**。
由允许的集合控制	针对集合检查此类型的字段，以确保其值被允许。
由策略控制	具有生成某个值的策略的条目提供以下功能： ● 生成值的机制，以及 ● 确保指定值属于允许值集合的机制。

CRI-O 具有以下默认能力列表，允许用于 pod 的每个容器：

- **CHOWN**
- **DAC_OVERRIDE**
- **FSETID**
- **FOWNER**

- **SETGID**
- **SETUID**
- **SETPCAP**
- **NET_BIND_SERVICE**
- **KILL**

容器使用此默认列表中的功能，但 Pod 清单作者可以通过请求额外功能或移除某些默认行为来修改列表。使用 **allowedCapabilities**、**defaultAddCapabilities** 和 **requiredDropCapabilities** 参数来控制来自容器集的此类请求。通过这些参数，您可以指定可以请求哪些功能，哪些必须添加到每一个容器，哪些必须被每个容器禁止或丢弃。

注意

您可以通过将 **requiredDropCapabilities** 参数设置为 **ALL** 来丢弃容器的所有功能。

15.1.3. 安全性上下文约束策略

RunAsUser

- **MustRunAs** - 需要配置 **runAsUser**。使用配置的 **runAsUser** 作为默认值。针对配置的 **runAsUser** 进行验证。

  MustRunAs 片断示例

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

- **MustRunAsRange** - 如果不使用预分配值，则需要定义最小值和最大值。使用最小值作为默认值。针对整个允许范围进行验证。

  MustRunAsRange 代码片段示例

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

- **MustRunAsNonRoot** - 需要 Pod 提交为具有非零 **runAsUser** 或具有镜像中定义的 **USER** 指令。不提供默认值。

  MustRunAsNonRoot 片断示例

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

```
...
```

- **RunAsAny** - 不提供默认值。允许指定任何 **runAsUser**。

  RunAsAny 代码片段示例

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

SELinuxContext

- **MustRunAs** - 如果不使用预分配的值，则需要配置 **seLinuxOptions**。使用 **seLinuxOptions** 作为默认值。针对 **seLinuxOptions** 进行验证。
- **RunAsAny** - 不提供默认值。允许指定任何 **seLinuxOptions**。

SupplementalGroups

- **MustRunAs** - 如果不使用预分配值，则需要至少指定一个范围。使用第一个范围内的最小值作为默认值。针对所有范围进行验证。
- **RunAsAny** - 不提供默认值。允许指定任何 **supplementalGroups**。

FSGroup

- **MustRunAs** - 如果不使用预分配值，则需要至少指定一个范围。使用第一个范围内的最小值作为默认值。针对第一个范围内的第一个 ID 进行验证。
- **RunAsAny** - 不提供默认值。允许指定任何 **fsGroup** ID。

15.1.4. 控制卷

通过设置 SCC 的 **volumes** 字段，控制特定卷类型的使用。此字段的允许值与创建卷时定义的卷来源对应：

- **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
- *（允许使用所有卷类型的一个特殊值）
- **none** （禁止使用所有卷类型的一个特殊值。仅为向后兼容而存在。）

为新 SCC 推荐的允许卷最小集合是
configMap、**downAPI**、**emptyDir**、**persistentVolumeClaim**、**secret** 和 **projected**。

注意

允许卷类型列表并不完整，因为每次发布新版 OpenShift Container Platform 时都会添加新的类型。

注意

为向后兼容，使用 **allowHostDirVolumePlugin** 将覆盖 **volumes** 字段中的设置。例如，如果 **allowHostDirVolumePlugin** 设为 false，但在 **volumes** 字段中是允许，则将移除 **volumes** 中的 **hostPath** 值。

15.1.5. 准入控制

利用 SCC 的*准入控制*可以根据授予用户的能力来控制资源的创建。

就 SCC 而言，这意味着准入控制器可以检查上下文中提供的用户信息以检索一组合适的 SCC。这样做可确保 Pod 具有相应的授权，能够提出与其操作环境相关的请求或生成一组要应用到 Pod 的约束。

准入用于授权 Pod 的 SCC 集合由用户身份和用户所属的组来决定。另外，如果 Pod 指定了服务帐户，则允许的 SCC 集合包括服务帐户可访问的所有约束。

准入使用以下方法来创建 Pod 的最终安全性上下文：

1. 检索所有可用的 SCC。
2. 为请求上未指定的安全性上下文设置生成字段值。
3. 针对可用约束来验证最终设置。

如果找到了匹配的约束集合，则接受 Pod。如果请求不能与 SCC 匹配，则拒绝 Pod。

Pod 必须针对 SCC 验证每一个字段。以下示例中只有其中两个字段必须验证：

注意

这些示例是在使用预分配值的策略上下文中。

FSGroup SCC 策略为 MustRunAs

如果 Pod 定义了 **fsGroup** ID，该 ID 必须等于默认的 **fsGroup** ID。否则，Pod 不会由该 SCC 验证，而会评估下一个 SCC。

如果 **SecurityContextConstraints.fsGroup** 字段的值为 **RunAsAny**，并且 Pod 规格省略了 **Pod.spec.securityContext.fsGroup**，则此字段被视为有效。注意在验证过程中，其他 SCC 设置可能会拒绝其他 Pod 字段，从而导致 Pod 失败。

SupplementalGroups SCC 策略为 MustRunAs

如果 Pod 规格定义了一个或多个 **supplementalGroups** ID，则 Pod 的 ID 必须等于命名空间的 **openshift.io/sa.scc.supplemental-groups** 注解中的某一个 ID。否则，Pod 不会由该 SCC 验证，而会评估下一个 SCC。

如果 **SecurityContextConstraints.supplementalGroups** 字段的值为 **RunAsAny**，并且 Pod 规格省略了 **Pod.spec.securityContext.supplementalGroups**，则此字段被视为有效。注意在验证过程中，其他 SCC 设置可能会拒绝其他 Pod 字段，从而导致 Pod 失败。

15.1.6. 安全性上下文约束优先级

安全性上下文约束 (SCC) 具有一个优先级字段，它会影响准入控制器尝试验证请求时的排序。

优先级值为 **0** 是最低优先级。nil 优先级被视为 **0** 或最低优先级。在排序时，优先级更高的 SCC 会被移到集合的前面。

确定了一组可用 SCC 后，SCC 会按照以下方式排序：

1. 最高优先级 SCC 首先排序。
2. 如果优先级相等，则 SCC 按照限制性最强到最弱排序。

3. 如果优先级和限制性都相等，则 SCC 按照名称排序。

默认情况下，授权给集群管理员的 **anyuid** SCC 在 SCC 集合中具有优先权。这允许集群管理员通过在 Pod 的 **SecurityContext** 中指定 **RunAsUser**，以任何用户身份运行 Pod。

15.2. 关于预分配安全性上下文约束值

准入控制器清楚安全性上下文约束 (SCC) 中的某些条件，这些条件会触发它从命名空间中查找预分配值并在处理 Pod 前填充 SCC。每个 SCC 策略都独立于其他策略进行评估，每个策略的预分配值（若为允许）与 Pod 规格值聚合，为运行中 Pod 中定义的不同 ID 生成最终值。

以下 SCC 导致准入控制器在 Pod 规格中没有定义范围时查找预分配值：

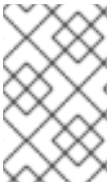
1. **RunAsUser** 策略为 **MustRunAsRange** 且未设置最小或最大值。准入查找 **openshift.io/sa.scc.uid-range** 注解来填充范围字段。
2. **SELinuxContext** 策略为 **MustRunAs** 且未设定级别。准入查找 **openshift.io/sa.scc.mcs** 注解来填充级别。
3. **FSGroup** 策略为 **MustRunAs**。准入查找 **openshift.io/sa.scc.supplemental-groups** 注解。
4. **SupplementalGroups** 策略为 **MustRunAs**。准入查找 **openshift.io/sa.scc.supplemental-groups** 注解。

在生成阶段，安全性上下文提供程序会对 Pod 中未具体设置的参数值使用默认值。默认值基于所选的策略：

1. **RunAsAny** 和 **MustRunAsNonRoot** 策略不提供默认值。如果 Pod 需要参数值，如组 ID，您必须在 Pod 规格中定义这个值。
2. **MustRunAs**（单值）策略提供始终使用的默认值。例如，对于组 ID，即使 Pod 规格定义了自己的 ID 值，命名空间的默认参数值也会出现在 Pod 的组中。
3. **MustRunAsRange** 和 **MustRunAs**（基于范围）策略提供范围的最小值。与单值 **MustRunAs** 策略一样，命名空间的默认参数值出现在运行的 Pod 中。如果基于范围的策略可以配置多个范围，它会提供配置的第一个范围的最小值。

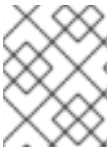
注意

如果命名空间上不存在 **openshift.io/sa.scc.supplemental-groups** 注解，则 **FSGroup** 和 **SupplementalGroups** 策略回退到 **openshift.io/sa.scc.uid-range** 注解。如果两者都不存在，则不创建 SCC。



注意

默认情况下，基于注解的 **FSGroup** 策略使用基于注解的最小值的单个范围来配置其自身。例如，如果您的注解显示为 **1/3**，则 **FSGroup** 策略使用最小值和最大值 **1** 配置其自身。如果要允许 **FSGroup** 字段接受多个组，可以配置不使用注解的自定义 SCC。



注意

openshift.io/sa.scc.supplemental-groups 注解接受以逗号分隔的块列表，格式为 **<start>/<length** 或 **<start>-<end>**。**openshift.io/sa.scc.uid-range** 注解只接受一个块。

15.3. 安全性上下文约束示例

以下示例演示了安全性上下文约束 (SCC) 格式和注解：

注解 privileged 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 Pod 可以请求的功能列表。空列表表示不允许请求任何功能，而特殊符号 * 则允许任何功能。

2 添加至任何 Pod 的附加功能列表。

3 **FSGroup** 策略，指明安全性上下文的允许值。

4 可访问此 SCC 的组。

5 从 pod 丢弃的功能列表。或者，指定 **ALL** 以丢弃所有功能。

6 **runAsUser** 策略类型，指明安全上下文的允许值。

7 **seLinuxContext** 策略类型，指明安全上下文的允许值。

8 **supplementalGroups** 策略，指明安全上下文的允许补充组。

9 可访问此 SCC 的用户。

SCC 中的 **users** 和 **groups** 字段控制能够访问该 SCC 的用户。默认情况下，集群管理员、节点和构建控制器被授予特权 SCC 的访问权限。所有经过身份验证的用户被授予受限 SCC 的访问权限。

无显式 runAsUser 设置

```
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 当容器或 Pod 没有指定应运行它的用户 ID 时，则生效的 UID 由发出此 Pod 的 SCC 决定。由于在默认情况下，受限 SCC 会授权给所有经过身份验证的用户，所以它可供所有用户和服务帐户使用，并在大多数情形中使用。受限 SCC 使用 **MustRunAsRange** 策略来约束并默认设定 **securityContext.runAsUser** 字段的可能值。准入插件会在当前项目上查找 **openshift.io/sa.scc.uid-range** 注解来填充范围字段，因为它不提供此范围。最后，容器的 **runAsUser** 值等于这一范围中的第一个值，而这难以预测，因为每个项目都有不同的范围。

带有显式 runAsUser 设置

```
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 只有服务帐户或用户被授予对允许某一用户 ID 的 SCC 访问权限时，OpenShift Container Platform 才会接受请求该用户 ID 的容器或 Pod。SCC 允许任意 ID、属于某一范围的 ID，或特定于请求的确切用户 ID。

此配置对 SELinux、fsGroup 和补充组有效。

15.4. 创建安全性上下文约束

您可以使用 OpenShift CLI（**oc**）创建安全性上下文约束（SCC）。

先决条件

- 安装 OpenShift CLI (**oc**) 。
- 以具有 **cluster-admin** 角色的用户身份登录集群。

流程

1. 在名为 **scc_admin.yaml** 的 YAML 文件中定义 SCC：

   SecurityContextConstraints 对象定义

```
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
```

   另外，您可以通过将 **requiredDropCapabilities** 字段设为所需的值来丢弃 SCC 的特定功能。所有指定的功能都会从容器中丢弃。要丢弃所有的能力，请指定 **ALL**。例如，要创建一个丢弃 **KILL**、**MKNOD** 和 **SYS_CHROOT** 功能的 SCC，请将以下内容添加到 SCC 对象中：

```
requiredDropCapabilities:
- KILL
- MKNOD
- SYS_CHROOT
```

注意

您不能列出 **allowedCapabilities** 和 **requiredDropCapabilities** 中的功能。

CRI-O 支持 Docker 文档中找到的相同功能值列表。

2. 通过传递文件来创建 SCC：

```
$ oc create -f scc_admin.yaml
```

输出示例

```
securitycontextconstraints "scc-admin" created
```

验证

- 验证 SCC 已创建好：

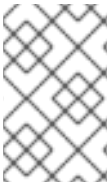
```
$ oc get scc scc-admin
```

输出示例

```
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. 基于角色的对安全性上下文限制的访问

您可以将 SCC 指定为由 RBAC 处理的资源。这样，您可以将对 SCC 访问的范围限定为某一项目或整个集群。直接将用户、组或服务帐户分配给 SCC 可保留整个集群的范围。



注意

您无法将 SCC 分配给在以下某一默认命名空间中创建的 Pod：**default**、**kube-system**、**kube-public**、**openshift-node**、**openshift-infra**、**openshift**。这些命名空间不应用于运行 pod 或服务。

要使您的角色包含对 SCC 的访问，请在创建角色时指定 **scc** 资源。

```
$ oc create role <role-name> --verb=use --resource=scc --resource-name=<scc-name> -n
<namespace>
```

这会生成以下角色定义：

```
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 角色的名称。

2 所定义角色的命名空间。若未指定，则默认为 **default**。

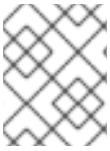
3 包含 **SecurityContextConstraints** 资源的 API 组。在 **scc** 指定为资源时自动定义。

4 要访问的 SCC 的示例名称。

5 允许用户在 **resourceNames** 字段中指定 SCC 名称的资源组名称。

6 应用到角色的操作动词列表。

当本地或集群角色具有这样的规则时，通过 RoleBinding 或 ClusterRoleBinding 与其绑定的主体可以使用用户定义的 SCC **scc-name**。

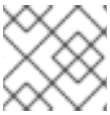


注意

由于 RBAC 旨在防止升级，因此即使项目管理员也无法授予 SCC 访问权限。默认情况下，不允许他们对 SCC 资源使用操作动词 **use**，包括 **restricted** SCC。

15.6. 安全性上下文约束命令参考

您可以使用 OpenShift CLI (**oc**) 将实例中的安全性上下文约束 (SCC) 作为常规 API 对象进行管理。



注意

您必须具有 **cluster-admin** 特权才能管理 SCC。

15.6.1. 列出安全性上下文约束

获取当前的 SCC 列表：

```
$ oc get scc
```

输出示例

```
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. 检查安全性上下文约束

您可以查看特定 SCC 的信息，包括这个 SCC 应用到哪些用户、服务帐户和组。

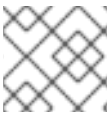
例如，检查 **restricted** SCC：

```
$ oc describe scc restricted
```

输出示例

```
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. 列出 SCC 应用到的用户和服务帐户。
2. 列出 SCC 应用到的组。



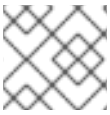
注意

要在升级过程中保留自定义 SCC，请不要编辑默认 SCC 的设置。

15.6.3. 删除安全性上下文约束

删除 SCC：

```
$ oc delete scc <scc_name>
```



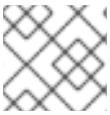
注意

如果删除了某一默认 SCC，重启集群时会重新生成该 SCC。

15.6.4. 更新安全性上下文约束

更新现有的 SCC：

```
$ oc edit scc <scc_name>
```



注意

要在升级过程中保留自定义 SCC，请不要编辑默认 SCC 的设置。

第 16 章 模拟 SYSTEM:ADMIN 用户

16.1. API 模仿

您可以配置对 OpenShift Container Platform API 的请求，使其表现为像是源自于另一用户。如需更多信息，请参阅 Kubernetes 文档中的用户模仿。

16.2. 模拟 SYSTEM:ADMIN 用户

您可以授予用户权限来模拟 **system:admin**，这将使它们获得集群管理员权限。

流程

- 要授予用户权限来模拟 **system:admin**，请运行以下命令：

```
$ oc create clusterrolebinding <any_valid_name> --clusterrole=sudoer --user=<username>
```

提示

您还可以应用以下 YAML 来授予模拟 **system:admin** 的权限：

```
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. 模拟 SYSTEM:ADMIN 组

当通过一个组为 **system:admin** 用户授予集群管理权限时，您必须在命令中包含 **--as=<user> --as-group=<group1> --as-group=<group2>** 参数来模拟关联的组。

流程

- 要通过模拟关联的集群管理组来模拟 **system:admin** 以为用户授予权限，请运行以下命令：

```
$ oc create clusterrolebinding <any_valid_name> --clusterrole=sudoer --as=<user> \
--as-group=<group1> --as-group=<group2>
```

第 17 章 同步 LDAP 组

作为管理员，您可以使用组来管理用户、更改其权限，并加强协作。您的组织可能已创建了用户组，并将其存储在 LDAP 服务器中。OpenShift Container Platform 可以将这些 LDAP 记录与 OpenShift Container Platform 内部记录同步，让您能够集中在一个位置管理您的组。OpenShift Container Platform 目前支持与使用以下三种通用模式定义组成员资格的 LDAP 服务器进行组同步：RFC 2307、Active Directory 和增强 Active Directory。

如需有关配置 LDAP 的更多信息，请参阅配置 LDAP 身份提供程序。

注意

您必须具有 **cluster-admin** 特权才能同步组。

17.1. 关于配置 LDAP 同步

在运行 LDAP 同步之前，您需要有一个同步配置文件。此文件包含以下 LDAP 客户端配置详情：

- 用于连接 LDAP 服务器的配置。
- 依赖于您的 LDAP 服务器中所用模式的同步配置选项。
- 管理员定义的名称映射列表，用于将 OpenShift Container Platform 组名称映射到 LDAP 服务器中的组。

配置文件的格式取决于您使用的模式，即 RFC 2307、Active Directory 或增强 Active Directory。

LDAP 客户端配置

配置中的 LDAP 客户端配置部分定义与 LDAP 服务器的连接。

配置中的 LDAP 客户端配置部分定义与 LDAP 服务器的连接。

LDAP 客户端配置

```
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) 连接协议、托管数据库的 LDAP 服务器的 IP 地址以及要连接的端口，格式为 **scheme://host:port**。

(2) 可选的可分辨名称 (DN)，用作绑定 DN。如果需要升级特权才能检索同步操作的条目，OpenShift Container Platform 会使用此项。

(3) 用于绑定的可选密码。如果需要升级特权才能检索同步操作的条目，OpenShift Container Platform 会使用此项。此值也可在环境变量、外部文件或加密文件中提供。

(4) 为 **false** 时，安全 LDAP **ldaps://** URL 使用 TLS 进行连接，并且不安全 LDAP **ldap://** URL 会被升级到 TLS。为 **true** 时，不会对服务器进行 TLS 连接，您不能使用 **ldaps://** URL。

(5) 用于验证所配置 URL 的服务器证书的证书捆绑包。如果为空，OpenShift Container Platform 将使用系统信任的根证书。只有 **insecure** 设为 **false** 时才会应用此项。

LDAP 查询定义

同步配置由用于同步所需条目的 LDAP 查询定义组成。LDAP 查询的具体定义取决于用来在 LDAP 服务器中存储成员资格信息的模式。

LDAP 查询定义

```
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. 所有搜索都应从其中开始的目录分支的可分辨名称 (DN)。您需要指定目录树的顶端，但也可以指定目录中的子树。
2. 搜索的范围。有效值为 **base**、**one** 或 **sub**。如果未定义，则假定为 **sub** 范围。下表中可找到范围选项的描述。
3. 与 LDAP 树中别名相关的搜索行为。有效值是 **never**、**search**、**base** 或 **always**。如果未定义，则默认为 **always** 解引用别名。下表中可找到有关解引用行为的描述。
4. 客户端可进行搜索的时间限值（以秒为单位）。**0** 代表不实施客户端限制。
5. 有效的 LDAP 搜索过滤器。如果未定义，则默认为 **(objectClass=*)**。
6. 服务器响应页面大小的可选最大值，以 LDAP 条目数衡量。如果设为 **0**，则不对响应页面实施大小限制。当查询返回的条目数量多于客户端或服务器默认允许的数量时，需要设置分页大小。

表 17.1. LDAP 搜索范围选项

LDAP 搜索范围	描述
base	仅考虑通过为查询给定的基本 DN 指定的对象。
one	考虑作为查询的基本 DN 的树中同一级上的所有对象。
sub	考虑根部是为查询给定的基本 DN 的整个子树。

表 17.2. LDAP 解引用行为

解引用行为	描述
never	从不解引用 LDAP 树中找到的任何别名。
search	仅解引用搜索时找到的别名。
base	仅在查找基本对象时解引用别名。

解引用行为	描述
always	始终解引用 LDAP 树中找到的所有别名。

用户定义的名称映射

用户定义的名称映射明确将 OpenShift Container Platform 组的名称映射到可在 LDAP 服务器上找到组的唯一标识符。映射使用普通 YAML 语法。用户定义的映射可为 LDAP 服务器中每个组包含一个条目，或者仅包含这些组的一个子集。如果 LDAP 服务器上的组没有用户定义的名称映射，同步过程中的默认行为是使用指定为 OpenShift Container Platform 组名称的属性。

用户定义的名称映射

```
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. 关于 RFC 2307 配置文件

RFC 2307 模式要求您提供用户和组条目的 LDAP 查询定义，以及在 OpenShift Container Platform 内部记录中代表它们的属性。

为明确起见，您在 OpenShift Container Platform 中创建的组应尽可能将可分辨名称以外的属性用于面向用户或管理员的字段。例如，通过电子邮件标识 OpenShift Container Platform 组的用户，并将该组的名称用作通用名称。以下配置文件创建了这些关系：

注意

如果使用用户定义的名称映射，您的配置文件会有所不同。

使用 RFC 2307 模式的 LDAP 同步配置：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. 存储该组记录的 LDAP 服务器的 IP 地址和主机。
2. 为 **false** 时，安全 LDAP **ldaps://** URL 使用 TLS 进行连接，并且不安全 LDAP **ldap://** URL 会被升级到 TLS。为 **true** 时，不会对服务器进行 TLS 连接，您不能使用 **ldaps://** URL。
3. 唯一标识 LDAP 服务器上组的属性。将 DN 用于 **groupUIDAttribute** 时，您无法指定 **groupsQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。
4. 要用作组名称的属性。
5. 存储成员资格信息的组属性。
6. 唯一标识 LDAP 服务器上用户的属性。将 DN 用于 userUIDAttribute 时，您无法指定**usersQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。
7. OpenShift Container Platform 组记录中用作用户名称的属性。

17.1.2. 关于 Active Directory 配置文件

Active Directory 模式要求您提供用户条目的 LDAP 查询定义，以及在内部 OpenShift Container Platform 组记录中代表它们的属性。

为明确起见，您在 OpenShift Container Platform 中创建的组应尽可能将可分辨名称以外的属性用于面向用户或管理员的字段。例如，通过电子邮件标识 OpenShift Container Platform 组的用户，但通过 LDAP 服务器上的组名称来定义组名称。以下配置文件创建了这些关系：

使用 Active Directory 模式的 LDAP 同步配置： 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 组记录中用作用户名称的属性。
2. 存储成员资格信息的用户属性。

17.1.3. 关于增强 Active Directory 配置文件

增强 Active Directory（augmented Active Directory） 模式要求您提供用户条目和组条目的 LDAP 查询定义，以及在内部 OpenShift Container Platform 组记录中代表它们的属性。

为明确起见，您在 OpenShift Container Platform 中创建的组应尽可能将可分辨名称以外的属性用于面向用户或管理员的字段。例如，通过电子邮件标识 OpenShift Container Platform 组的用户，并将该组的名称用作通用名称。以下配置文件创建了这些关系。

使用增强 Active Directory 模式的 LDAP 同步配置： increaseded_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. 唯一标识 LDAP 服务器上组的属性。将 DN 用于 groupUIDAttribute 时，您无法指定 **groupsQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。
2. 要用作组名称的属性。
3. OpenShift Container Platform 组记录中用作用户名称的属性。
4. 存储成员资格信息的用户属性。

17.2. 运行 LDAP 同步

创建了同步配置文件后，您可以开始同步。OpenShift Container Platform 允许管理员与同一服务器进行多种不同类型的同步。

17.2.1. 将 LDAP 服务器与 OpenShift Container Platform 同步

您可以将 LDAP 服务器上的所有组与 OpenShift Container Platform 同步。

先决条件

- 创建同步配置文件。

流程

- 将 LDAP 服务器上的所有组与 OpenShift Container Platform 同步：

```
$ oc adm groups sync --sync-config=config.yaml --confirm
```

注意

默认情况下，所有组同步操作都是空运行（dry-run），因此您必须在 **oc adm groups sync** 命令上设置 **--confirm** 标志，才能更改 OpenShift Container Platform 组记录。

17.2.2. 将 OpenShift Container Platform 组与 LDAP 服务器同步

您可以同步所有已在 OpenShift Container Platform 中并与配置文件中指定的 LDAP 服务器中的组相对应的组。

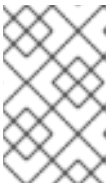
先决条件

- 创建同步配置文件。

流程

- 将 OpenShift Container Platform 组与 LDAP 服务器同步：

```
$ oc adm groups sync --type=openshift --sync-config=config.yaml --confirm
```



注意

默认情况下，所有组同步操作都是空运行（dry-run），因此您必须在 **oc adm groups sync** 命令上设置 **--confirm** 标志，才能更改 OpenShift Container Platform 组记录。

17.2.3. 将 LDAP 服务器上的子组与 OpenShift Container Platform 同步

您可以使用白名单文件和/或黑名单文件，将 LDAP 组的子集与 OpenShift Container Platform 同步。

注意

您可以使用黑名单文件、白名单文件或白名单字面量的任意组合。白名单和黑名单文件必须每行包含一个唯一组标识符，您可以在该命令本身中直接包含白名单字面量。这些准则适用于在 LDAP 服务器上找到的组，以及 OpenShift Container Platform 中已存在的组。

先决条件

- 创建同步配置文件。

流程

- 要将 LDAP 组的子集与 OpenShift Container Platform 同步，请使用以下任一命令：

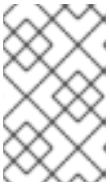
```
$ oc adm groups sync --whitelist=<whitelist_file> \
                     --sync-config=config.yaml     \
                     --confirm
```

```
$ 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
```



注意

默认情况下，所有组同步操作都是空运行（dry-run），因此您必须在 **oc adm groups sync** 命令上设置 **--confirm** 标志，才能更改 OpenShift Container Platform 组记录。

17.3. 运行组修剪任务

如果创建组的 LDAP 服务器上的记录已不存在，管理员也可以选择从 OpenShift Container Platform 记录中移除这些组。修剪任务将接受用于同步任务的相同同步配置文件以及白名单或黑名单。

例如：

```
$ 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. 自动同步 LDAP 组

您可以通过配置 cron 作业来定期自动同步 LDAP 组。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 您已配置了 LDAP 身份提供程序 (IDP)。
  此流程假设您创建一个名为 **ldap-secret** 的 LDAP secret 和名为 **ca-config-map** 的配置映射。

流程

1. 创建一个在其中运行 cron 任务的项目：

```
$ oc new-project ldap-sync 1
```

1 此流程使用名为 **ldap-sync** 的项目。

2. 找到您在配置 LDAP 身份提供程序时创建的 secret 和配置映射，并将它们复制到此新项目。secret 和配置映射存在于 **openshift-config** 项目中，必须复制到新的 **ldap-sync** 项目中。

3. 定义服务帐户：

ldap-sync-service-account.yaml示例

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
```

4. 创建服务帐户：

```
$ oc create -f ldap-sync-service-account.yaml
```

5. 定义集群角色：

ldap-sync-cluster-role.yaml示例

```
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. 创建集群角色：

```
$ oc create -f ldap-sync-cluster-role.yaml
```

7. 定义集群角色绑定将集群角色绑定绑定到服务帐户：

ldap-sync-cluster-role-binding.yaml 示例

```
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 对此过程前面创建的服务帐户的引用。

2 引用此流程前面创建的集群角色。

8. 创建集群角色绑定：

```
$ oc create -f ldap-sync-cluster-role-binding.yaml
```

9. 定义指定同步配置文件的配置映射：

ldap-sync-config-map.yaml 示例

```
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. 定义同步配置文件。
2. 指定 URL。
3. 指定 **bindDN**。
4. 本例使用 RFC2307 模式；根据需要调整值。您还可以使用不同的架构。
5. 为 **groupsQuery** 指定 **baseDN**。
6. 为 **usersQuery** 指定 **baseDN**。

10. 创建配置映射：

```
$ oc create -f ldap-sync-config-map.yaml
```

11. 定义 cron 作业：

ldap-sync-cron-job.yaml 示例

```
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. 配置 cron 作业的设置。有关 cron 作业设置的更多信息，请参阅"创建 cron 作业"。
2. 以 cron 格式指定的作业计划。这个示例 cron 作业每 30 分钟运行一次。根据需要调整频率，确保考虑到同步运行所需的时间。
3. 维护完成的作业的时间（以秒为单位）。这应该与作业调度的期限匹配，以便清理旧的失败作业并防止不必要的警报。如需更多信息，请参阅 Kubernetes 文档中的 TTL-after-finished Controller。
4. 运行 cron 作业的 LDAP sync 命令。传递配置映射中定义的同步配置文件。
5. 此 secret 是在配置 LDAP IDP 时创建的。
6. 此配置映射是在配置 LDAP IDP 时创建的。

12. 创建 cron job：

```
$ oc create -f ldap-sync-cron-job.yaml
```

其他资源

- 配置 LDAP 身份提供程序
- 创建 cron job

17.5. LDAP 组同步示例

本节包含 RFC 2307、Active Directory 和增强 Active Directory 模式的示例。

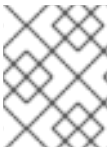
注意

这些示例假定所有用户都是其各自组的直接成员。具体而言，没有任何组的成员是其他组。如需有关如何同步嵌套组的信息，请参见嵌套成员资格同步示例。

17.5.1. 使用 RFC 2307 模式同步组

对于 RFC 2307 模式，以下示例将同步名为 **admins** 的组，该组有两个成员 **Jane** 和 **Jim**。这些示例阐述了：

- 如何将组和用户添加到 LDAP 服务器中。
- 同步之后 OpenShift Container Platform 中会生成什么组记录。



注意

这些示例假定所有用户都是其各自组的直接成员。具体而言，没有任何组的成员是其他组。如需有关如何同步嵌套组的信息，请参见嵌套成员资格同步示例。

在 RFC 2307 模式中，用户（Jane 和 Jim）和组都作为第一类条目存在于 LDAP 服务器上，组成员资格则存储在组的属性中。以下 **ldif** 片段定义了这个模式的用户和组：

使用 RFC 2307 模式的 LDAP 条目：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. 组是 LDAP 服务器中的第一类条目。
2. 组成员使用作为组属性的标识引用来列出。

先决条件

- 创建配置文件。

流程

- 使用 **rfc2307_config.yaml** 文件运行同步：

```
$ oc adm groups sync --sync-config=rfc2307_config.yaml --confirm
```

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 rfc2307_config.yaml 文件创建的 OpenShift Container Platform 组

```
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) 此 OpenShift Container Platform 组与 LDAP 服务器最后一次同步的时间，采用 ISO 6801 格式。

(2) LDAP 服务器上组的唯一标识符。

(3) 存储该组记录的 LDAP 服务器的 IP 地址和主机。

(4) 根据同步文件指定的组的名称。

(5) 属于组的成员的用户，名称由同步文件指定。

17.5.2. 使用 RFC2307 模式及用户定义的名称映射来同步组

使用用户定义的名称映射同步组时，配置文件会更改为包含这些映射，如下所示。

使用 RFC 2307 模式及用户定义的名称映射的 LDAP 同步配置：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 用户定义的名称映射。

2 唯一标识符属性，用于用户定义的名称映射中的键。将 DN 用于 groupUIDAttribute 时，您无法指定 **groupsQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。

3 如果其唯一标识符不在用户定义的名称映射中，用于指定 OpenShift Container Platform 组的属性。

4 唯一标识 LDAP 服务器上用户的属性。将 DN 用于 userUIDAttribute 时，您无法指定**usersQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。

先决条件

- 创建配置文件。

流程

- 使用 **rfc2307_config_user_defined.yaml** 文件运行同步：

```
$ oc adm groups sync --sync-config=rfc2307_config_user_defined.yaml --confirm
```

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 rfc2307_config_user_defined.yaml 文件创建的 OpenShift Container Platform 组

```
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 由用户定义的名称映射指定的组名称。

17.5.3. 使用 RFC 2307 及用户定义的容错来同步组

默认情况下，如果要同步的组包含其条目在成员查询中定义范围之外的成员，组同步会失败并显示以下错误：

```
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>")".
```

这通常表示 **usersQuery** 字段中的 **baseDN** 配置错误。不过，如果 **baseDN** 有意不含有组中的部分成员，那么设置 **tolerateMemberOutOfScopeErrors: true** 可以让组同步继续进行。范围之外的成员将被忽略。

同样，当组同步过程未能找到某个组的某一成员时，它会彻底失败并显示错误：

```
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".
```

这通常表示 **usersQuery** 字段配置错误。不过，如果组中包含已知缺失的成员条目，那么设置 **tolerateMemberNotFoundErrors: true** 可以让组同步继续进行。有问题的成员将被忽略。

警告

为 LDAP 组同步启用容错会导致同步过程忽略有问题的成员条目。如果 LDAP 组同步配置不正确，这可能会导致同步的 OpenShift Container Platform 组中缺少成员。

使用 RFC 2307 模式并且组成员资格有问题的 LDAP 条目：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 服务器上不存在的成员。
2. 可能存在，但不在同步任务的用户查询的 **baseDN** 下的成员。

要容许以上示例中的错误，您必须在同步配置文件中添加以下内容：

使用 RFC 2307 模式且容许错误的 LDAP 同步配置：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. 唯一标识 LDAP 服务器上用户的属性。将 DN 用于 userUIDAttribute 时，您无法指定**usersQuery** 过滤器。若要进行精细过滤，请使用白名单/黑名单方法。
2. 为 **true** 时，同步任务容许找不到部分成员的组，并且找不到 LDAP 条目的成员将被忽略。如果找不到组成员，同步任务的默认行为将失败。
3. 为 **true** 时，同步任务容许其部分成员在 **usersQuery** 基本 DN 中给定用户范围之外的组，并且不在成员查询范围的成员将被忽略。如果组中某个成员超出范围，则同步任务的默认行为将失败。

先决条件

- 创建配置文件。

流程

- 使用 **rfc2307_config_tolerating.yaml** 文件运行同步：

```
$ oc adm groups sync --sync-config=rfc2307_config_tolerating.yaml --confirm
```

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 rfc2307_config.yaml 文件创建的 OpenShift Container Platform 组

```
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) 属于组的成员的用户，根据同步文件指定。缺少查询遇到容许错误的成员。

17.5.4. 使用 Active Directory 模式同步组

在 Active Directory 模式中，两个用户（Jane 和 Jim）都作为第一类条目存在于 LDAP 服务器中，组成员资格则存储在用户的属性中。以下 **ldif** 片段定义了这个模式的用户和组：

使用 Active Directory 模式的 LDAP 条目： 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 用户的组成员资格列为用户的属性，组没有作为条目存在于服务器中。**memberOf** 属性不一定是用户的字面量属性；在一些 LDAP 服务器中，它在搜索过程中创建并返回给客户端，但不提交给数据库。

先决条件

- 创建配置文件。

流程

- 使用 **active_directory_config.yaml** 文件运行同步：

```
$ oc adm groups sync --sync-config=active_directory_config.yaml --confirm
```

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 active_directory_config.yaml 文件创建的 OpenShift Container Platform 组

```
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 此 OpenShift Container Platform 组与 LDAP 服务器最后一次同步的时间，采用 ISO 6801 格式。

2 LDAP 服务器上组的唯一标识符。

3 存储该组记录的 LDAP 服务器的 IP 地址和主机。

4 LDAP 服务器中列出的组名称。

5 属于组的成员的用户，名称由同步文件指定。

17.5.5. 使用增强 Active Directory 模式同步组

在增强 Active Directory 模式中，用户（Jane 和 Jim）和组都作为第一类条目存在于 LDAP 服务器中，组成员资格则存储在用户的属性中。以下 **ldif** 片段定义了这个模式的用户和组：

使用增强 Active Directory 模式的 LDAP 条目： increaseded_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) 用户的组成员资格列为用户的属性。

(2) 组是在 LDAP 服务器上的第一类条目。

先决条件

- 创建配置文件。

流程

- 使用 **augmented_active_directory_config.yaml** 文件运行同步：

```
$ oc adm groups sync --sync-config=augmented_active_directory_config.yaml --confirm
```

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 augmented_active_directory_config.yaml 文件创建的 OpenShift Container Platform 组

```
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) 此 OpenShift Container Platform 组与 LDAP 服务器最后一次同步的时间，采用 ISO 6801 格式。

(2) LDAP 服务器上组的唯一标识符。

(3) 存储该组记录的 LDAP 服务器的 IP 地址和主机。

(4) 根据同步文件指定的组的名称。

(5) 属于组的成员的用户，名称由同步文件指定。

17.5.5.1. LDAP 嵌套成员资格同步示例

OpenShift Container Platform 中的组不嵌套。在消耗数据之前，LDAP 服务器必须平展组成员资格。Microsoft 的 Active Directory Server 通过 **LDAP_MATCHING_RULE_IN_CHAIN** 规则支持这一功能，其 OID 为 **1.2.840.113556.1.4.1941**。另外，使用此匹配规则时只能同步明确列在白名单中的组。

本节中的示例使用了增强 Active Directory 模式，它将同步一个名为 **admins** 的组，该组有一个用户 **Jane** 和一个组 **otheradmins**。**otheradmins** 组具有一个用户成员：**Jim**。这个示例阐述了：

- 如何将组和用户添加到 LDAP 服务器中。
- LDAP 同步配置文件的概貌。
- 同步之后 OpenShift Container Platform 中会生成什么组记录。

在增强 Active Directory 模式中，用户（**Jane** 和 **Jim**）和组都作为第一类条目存在于 LDAP 服务器中，组成员资格则存储在用户或组的属性中。以下 **ldif** 片段定义了这个模式的用户和组：

使用增强 Active Directory 模式和嵌套成员的 LDAP 条目：increaseded_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) 用户和组的成员资格列为对象的属性。

(3) (4) 组是在 LDAP 服务器上的第一类条目。

(6) **otheradmins** 组是 **admins** 组的成员。

与 Active Directory 同步嵌套的组时，您必须提供用户条目和组条目的 LDAP 查询定义，以及在内部 OpenShift Container Platform 组记录中代表它们的属性。另外，此配置也需要进行某些修改：

- **oc adm groups sync** 命令必须明确将组列在白名单中。
- 用户的 **groupMembershipAttributes** 必须包含 **"memberOf:1.2.840.113556.1.4.1941:"**，以遵守 **LDAP_MATCHING_RULE_IN_CHAIN** 规则。

- **groupUIDAttribute** 必须设为 **dn**。
- **groupsQuery**：
  - 不得设置 **filter**。
  - 必须设置有效的 **derefAliases**。
  - 不应设置 **basedn**，因为此值将被忽略。
  - 不应设置 **scope**，因为此值将被忽略。

为明确起见，您在 OpenShift Container Platform 中创建的组应尽可能将可分辨名称以外的属性用于面向用户或管理员的字段。例如，通过电子邮件标识 OpenShift Container Platform 组的用户，并将该组的名称用作通用名称。以下配置文件创建了这些关系：

使用增强 Active Directory 模式和嵌套成员的 LDAP 同步配置：increaseded_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) 无法指定 **groupsQuery** 过滤器。**groupsQuery** 基本 DN 和范围值将被忽略。**groupsQuery** 必需设置为一个有效的 **derefAliases**。

(2) 唯一标识 LDAP 服务器上组的属性。必须设为 **dn**。

(3) 要用作组名称的属性。

(4) OpenShift Container Platform 组记录中用作用户名称的属性。多数安装中首选**mail** 或 **sAMAccountName**。

(5) 存储成员资格信息的用户属性。注意 **LDAP_MATCHING_RULE_IN_CHAIN** 的使用。

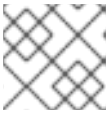
先决条件

- 创建配置文件。

流程

- 使用 **augmented_active_directory_config_nested.yaml** 文件运行同步：

```
$ oc adm groups sync \
    'cn=admins,ou=groups,dc=example,dc=com' \
    --sync-config=augmented_active_directory_config_nested.yaml \
    --confirm
```



注意

必须明确将 **cn=admins,ou=groups,dc=example,dc=com** 组列在白名单中。

OpenShift Container Platform 创建以下组记录作为上述同步操作的结果：

使用 augmented_active_directory_config_nested.yaml 文件创建的 OpenShift Container Platform 组

```
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) 此 OpenShift Container Platform 组与 LDAP 服务器最后一次同步的时间，采用 ISO 6801 格式。

(2) LDAP 服务器上组的唯一标识符。

(3) 存储该组记录的 LDAP 服务器的 IP 地址和主机。

(4) 根据同步文件指定的组的名称。

(5) 属于组的成员的用户，名称由同步文件指定。请注意，嵌套组成员包含在内，因为 Microsoft Active Directory Server 已经平展了组成员关系。

17.6. LDAP 同步配置规格

配置文件的对象规格如下。请注意，不同的模式对象有不同的字段。例如，v1.ActiveDirectoryConfig 没有 **groupsQuery** 字段，而 v1.RFC2307Config 和 v1.AugmentedActiveDirectoryConfig 都有这个字段。

重要

不支持二进制属性。所有来自 LDAP 服务器的属性数据都必须采用 UTF-8 编码字符串的格式。例如，切勿将 **objectGUID** 等二进制属性用作 ID 属性。您必须改为使用字符串属性，如 **sAMAccountName** 或 **userPrincipalName**。

17.6.1. v1.LDAPSyncConfig

LDAPSyncConfig 包含定义 LDAP 组同步所需的配置选项。

名称	描述	模式
kind	代表此对象所代表的 REST 资源的字符串值。服务器可以从客户端向其提交请求的端点推断。无法更新。采用驼峰拼写法 (CamelCase)。更多信息：https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#types-kinds	字符串
apiVersion	定义对象的此表示法的版本控制模式。服务器应该将识别的模式转换为最新的内部值，并可拒绝未识别的值。更多信息：https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#resources	字符串
url	主机是要连接到的 LDAP 服务器的方案、主机和端口：**scheme://host:port**	字符串
bindDN	要绑定到 LDAP 服务器的可选 DN。	字符串
bindPassword	在搜索阶段要绑定的可选密码。	v1.StringSource
insecure	若为 **true**，则表示连接不应使用 TLS。若为 **false**，则 **ldaps://** URL 使用 TLS 进行连接，并且使用 https://tools.ietf.org/html/rfc2830 中指定的 StartTLS 将 **ldap://** URL 升级为 TLS 连接。如果将 **insecure** 设为 **true**，则无法使用 **ldaps://** URL。	布尔值
ca	在向服务器发出请求时使用的可选的可信证书颁发机构捆绑包。若为空，则使用默认的系统根证书。	字符串

名称	描述	模式
groupUIDNameMapping	LDAP 组 UID 与 OpenShift Container Platform 组名称的可选直接映射。	对象
rfc2307	包含用于从设置的 LDAP 服务器提取数据的配置，其格式类似于 RFC2307：第一类组和用户条目，以及由列出其成员的组条目的多值属性决定的组成员资格。	v1.RFC2307Config
activeDirectory	包含用于从设置的 LDAP 服务器提取数据的配置，其格式与 Active Directory 中使用的类似：第一类用户条目，以及由列出所在组的成员的多值属性决定的组成员资格。	v1.ActiveDirectoryConfig
augmentedActiveDirectory	包含用于从设置的 LDAP 服务器提取数据的配置，其格式与上面描述的 Active Directory 中使用的类似，再另加一项：有第一类组条目，它们用来保存元数据而非组成员资格。	v1.AugmentedActiveDirectoryConfig

17.6.2. v1.StringSource

StringSource 允许指定内联字符串，或通过环境变量或文件从外部指定。当它只包含一个字符串值时，它会编列为一个简单 JSON 字符串。

名称	描述	模式
value	指定明文值，或指定加密值（如果指定了 **keyFile**）。	字符串
env	指定包含明文值或加密值（如果指定了 **keyFile**）的环境变量。	字符串
file	引用含有明文值或加密值（如果指定了 **keyFile**）的文件。	字符串
keyFile	引用包含用于解密值的密钥的文件。	字符串

17.6.3. v1.LDAPQuery

LDAPQuery 包含构建 LDAP 查询时所需的选项。

名称	描述	模式
baseDN	所有搜索都应从中开始的目录分支的 DN。	字符串
scope	搜索的可选范围。可以是 **base**（仅基本对象）、**one**（基本级别上的所有对象）或 **sub**（整个子树）。若未设置，则默认为 **sub**。	字符串
derefAliases	与别名相关的可选搜索行为。可以是 **never**（从不解引用别名）、**search**（仅在搜索中解引用）、**base**（仅在查找基本对象时解引用）或 **always**（始终解引用）。若未设置，则默认为 **always**。	字符串
timeout	包含所有对服务器的请求在放弃等待响应前应保持待定的时间限制，以秒为单位。如果是 **0**，则不会实施客户端一侧的限制。	整数
filter	使用基本 DN 从 LDAP 服务器检索所有相关条目的有效 LDAP 搜索过滤器。	字符串
pageSize	最大首选页面大小，以 LDAP 条目数衡量。页面大小为 **0** 表示不进行分页。	整数

17.6.4. v1.RFC2307Config

RFC2307Config 包含必要的配置选项，用于定义 LDAP 组同步如何使用 RFC2307 模式与 LDAP 服务器交互。

名称	描述	模式
groupsQuery	包含用于返回组条目的 LDAP 查询模板。	v1.LDAPQuery
groupUIDAttribute	定义 LDAP 组条目上的哪个属性将解释为其唯一标识符。(**ldapGroupUID**)	字符串
groupNameAttributes	定义 LDAP 组条目上的哪些属性将解释为用于 OpenShift Container Platform 组的名称。	字符串数组

名称	描述	模式
groupMembershipAttributes	定义 LDAP 组条目上哪些属性将解释为其成员。这些属性中包含的值必须可由 **UserUIDAttribute** 查询。	字符串数组
usersQuery	包含用于返回用户条目的 LDAP 查询模板。	v1.LDAPQuery
userUIDAttribute	定义 LDAP 用户条目上的哪个属性将解释为其唯一标识符。它必须与 **GroupMembershipAttributes** 中找到的值对应。	字符串
userNameAttributes	定义要使用 LDAP 用户条目上的哪些属性，以用作其 OpenShift Container Platform 用户名。使用第一个带有非空值的属性。这应该与您的 **LDAPPasswordIdentityProvider** 的 **PreferredUsername** 设置匹配。OpenShift Container Platform 组记录中用作用户名称的属性。多数安装中首选 **mail** 或 **sAMAccountName**。	字符串数组
tolerateMemberNotFoundErrors	决定在遇到缺失的用户条目时 LDAP 同步任务的行为。若为 **true**，则容许找不到任何匹配项的 LDAP 用户查询，并且只记录错误。若为 **false**，则 LDAP 同步任务在用户查询找不到匹配项时将失败。默认值为 **false**。如果此标志设为 **true**，则配置错误的 LDAP 同步任务可导致组成员资格被移除，因此建议谨慎使用此标志。	布尔值
tolerateMemberOutOfScopeErrors	决定在遇到超出范围的用户条目时 LDAP 同步任务的行为。如果为 **true**，则容许超出为所有用户查询给定的基本 DN 范围的 LDAP 用户查询，并且仅记录错误。若为 **false**，则当用户查询在所有用户查询指定的基本 DN 范围外搜索时，LDAP 同步任务将失败。如果此标志设为 **true**，则配置错误的 LDAP 同步任务可导致组中缺少用户，因此建议谨慎使用此标志。	布尔值

17.6.5. v1.ActiveDirectoryConfig

ActiveDirectoryConfig 包含必要的配置选项，用于定义 LDAP 组同步如何使用 Active Directory 模式与 LDAP 服务器交互。

名称	描述	模式
usersQuery	包含用于返回用户条目的 LDAP 查询模板。	v1.LDAPQuery
userNameAttributes	定义 LDAP 用户条目上的哪些属性将解释为其 OpenShift Container Platform 用户名。OpenShift Container Platform 组记录中用作用户名称的属性。多数安装中首选**mail** 或 **sAMAccountName**。	字符串数组
groupMembershipAttributes	定义 LDAP 用户条目上的哪些属性将解释为它所属的组。	字符串数组

17.6.6. v1.AugmentedActiveDirectoryConfig

AugmentedActiveDirectoryConfig 包含必要的配置选项，用于定义 LDAP 组同步如何使用增强 Active Directory 模式与 LDAP 服务器交互。

名称	描述	模式
usersQuery	包含用于返回用户条目的 LDAP 查询模板。	v1.LDAPQuery
userNameAttributes	定义 LDAP 用户条目上的哪些属性将解释为其 OpenShift Container Platform 用户名。OpenShift Container Platform 组记录中用作用户名称的属性。多数安装中首选**mail** 或 **sAMAccountName**。	字符串数组
groupMembershipAttributes	定义 LDAP 用户条目上的哪些属性将解释为它所属的组。	字符串数组
groupsQuery	包含用于返回组条目的 LDAP 查询模板。	v1.LDAPQuery
groupUIDAttribute	定义 LDAP 组条目上的哪个属性将解释为其唯一标识符。(**ldapGroupUID**)	字符串
groupNameAttributes	定义 LDAP 组条目上的哪些属性将解释为用于 OpenShift Container Platform 组的名称。	字符串数组

第 18 章 管理云供应商凭证

18.1. 关于 CLOUD CREDENTIAL OPERATOR

Cloud Credential Operator（CCO） 将云供应商凭证作为自定义资源定义 (CRD) 进行管理。**CredentialsRequest** 自定义资源（CR）的 CCO 同步，允许 OpenShift Container Platform 组件使用集群运行所需的特定权限请求云供应商凭证。

通过在 **install-config.yaml** 文件中为 **credentialsMode** 参数设置不同的值，可将 CCO 配置为以几种不同模式操作。如果没有指定模式，或将 **credentialsMode** 参数被设置为空字符串（""）。

18.1.1. 模式

通过在 **install-config.yaml** 文件中为 **credentialsMode** 参数设置不同的值，可将 CCO 配置为在 *mint*、*passthrough* 或 *manual* 模式下操作。这些选项为 CCO 使用云凭证处理集群中的 **credentialsRequest** CR 提供了透明性和灵活性，并允许配置 CCO 以适应您的机构的安全要求。不是所有 CCO 模式都支持所有云供应商。

- Mint：在 mint 模式中，CCO 使用提供的管理员级云凭证为集群中组件创建新凭证，且只具有所需的特定权限。

  注意

  Mint 模式是 CCO 的默认和最佳实践设置。

- Passthrough：在 passthrough 模式中，CCO 将提供的云凭证传递给请求云凭证的组件。
- Manual：在手动模式中，用户管理云凭证而不是 CCO。
  - 使用 AWS STS 手动：在手动模式中，您可以将 AWS 集群配置为使用 Amazon Web Services Secure Token Service（AWS STS）。借助这一配置，CCO 对不同组件使用临时凭证。

表 18.1. CCO 模式支持列表

云供应商	Mint	Passthrough	Manual（手动）
Amazon Web Services (AWS)	X	X	X
Microsoft Azure		X	X
Google Cloud Platform (GCP)	X	X	X
Red Hat OpenStack Platform(RHOSP)		X	
Red Hat Virtualization（RHV）		X	
VMware vSphere		X	

18.1.2. 默认行为

对于支持多个模式的平台（AWS、Azure 和 GCP），当 CCO 采用默认模式运行时，它会动态检查提供的凭证，以确定它们足以处理 **credentialsRequest** CR 的模式。

默认情况下，CCO 决定凭证是否足以满足 mint 模式（首选操作模式），并使用这些凭证为集群中组件创建适当的凭证。如果凭据不足以满足 mint 模式，它会决定凭证是否足以满足 passthrough 模式。如果凭据不足以满足 passthrough 模式，则 CCO 无法正确处理 **CredentialsRequest** CR。

如果确定提供的凭证在安装过程中不足，安装会失败。对于 AWS，安装程序在进程早期失败，并指示缺少哪些所需权限。在遇到错误之前，其他供应商可能不会提供有关错误原因的具体信息。

如果在安装成功后修改凭证，并且 CCO 确定新凭证不足，CCO 会给任何新的 **CredentialsRequest** CR 设置条件，表示因为凭证不足而无法处理这些凭证。

要解决凭据不足的问题，请为凭证提供足够权限。如果在安装过程中出现错误，请尝试再次安装。对于新 **CredentialsRequest** CR 的问题，请等待 CCO 尝试再次处理 CR。另外，您可以为 AWS、Azure 和 GCP 手动创建 IAM。

18.1.3. 其他资源

- Cloud Credential Operator 的 Cluster Operators 参考页面

18.2. 使用 MINT 模式

Amazon Web Services(AWS)和 Google Cloud Platform(GCP)支持 Mint 模式。

Mint 模式是支持它的平台的默认模式。在这个模式下，Cloud Credential Operator（CCO） 使用提供的管理员级云凭证为集群中组件创建新凭证，且只具有所需的特定权限。

如果在安装后没有删除凭证，则会存储并供 CCO 使用来处理集群中组件的 **CredentialsRequest** CR，并为每个组件创建新凭证，每个凭证只具有所需的特定权限。以 mint 模式持续协调云凭证可进行需要额外凭证或权限（如升级）的操作。

Mint 模式将管理员级别的凭证存储在集群 **kube-system** 命名空间中。如果此方法不符合您的机构的安全要求，请参阅 AWS 或 GCP 的 *将管理员级别的 secret 存储在 kube-system 项目中的替代方法*。

18.2.1. Mint 模式权限要求

以 mint 模式使用 CCO 时，请确保您提供的凭证满足运行或安装 OpenShift Container Platform 的云要求。如果提供的凭证不足以满足 mint 模式，则 CCO 无法创建 IAM 用户。

18.2.1.1. Amazon Web Services（AWS）权限

您在 AWS 中为 mint 模式提供的凭证必须具有以下权限：

- **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. Google Cloud Platform（GCP）权限

您在 GCP 中为 mint 模式提供的凭证必须具有以下权限：

- **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. 管理凭证 root secret 格式

每个云供应商都使用 **kube-system** 命名空间中的一个凭证 root secret，用于满足所有凭证请求并创建它们对应的 secret。这可以通过 mint 新凭证使用 *mint 模式* 完成，或使用 *passthrough 模式* 复制凭证 root secret。

secret 的格式因云而异，也用于每个 **CredentialsRequest** secret。

Amazon Web Services(AWS)secret 格式

```
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)secret 格式

```
apiVersion: v1
```

```
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

18.2.3. 带有删除或轮转管理员级别的凭证的 Mint 模式

目前，只有 AWS 和 GCP 支持这个模式。

在这个模式中，用户使用管理员级别的凭证安装 OpenShift Container Platform，就像正常的 mint 模式一样。但是，这个过程会在安装后从集群中删除管理员级别的凭证 secret。

管理员可以让 Cloud Credential Operator 自行请求只读凭证，许它验证所有 **CredentialsRequest** 对象是否有其所需的权限。因此，除非需要更改内容，否则不需要管理员级别的凭证。删除关联的凭证后，可以根据需要在底层云中删除或取消激活它。

注意

在非 z-stream 升级前，您必须使用管理员级别的凭证重新恢复凭证 secret。如果没有凭证，则可能会阻止升级。

管理员级别的凭证不会永久存储在集群中。

按照以下步骤，在短时间内仍然需要集群中的管理员级别的凭证。它还需要手动使用每次升级的管理员级别的凭证重新启用 secret。

18.2.3.1. 手动轮转云供应商凭证

如果因为某种原因更改了云供应商凭证，您必须手动更新 Cloud Credential Operator（CCO）用来管理云供应商凭证的 secret。

轮转云凭证的过程取决于 CCO 配置使用的模式。在为使用 mint 模式的集群轮转凭证后，您必须手动删除由删除凭证创建的组件凭证。

先决条件

- 您的集群会在支持使用您要使用的 CCO 模式手动轮转云凭证的平台上安装：
  - 对于 mint 模式，支持 Amazon Web Services (AWS)和 Google Cloud Platform (GCP)。
- 您已更改了用于与云供应商接口的凭证。
- 新凭证有足够的权限来在集群中使用 CCO 模式。

流程

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Workloads → Secrets**。
2. 在 **Secrets** 页面的表中，找到您的云供应商的 root secret。

平台	Secret 名称
AWS	**aws-creds**
GCP	**gcp-credentials**

3. 点击与 secret 相同的行  中的 **Options** 菜单，然后选择 **Edit Secret**。
4. 记录 **Value** 字段的内容。您可以使用这些信息验证在更新凭证后该值是否不同。
5. 使用云供应商的新身份验证信息更新 **Value** 字段的文本，然后点 **Save**。
6. 如果集群的 CCO 配置为使用 mint 模式，请删除各个 **CredentialsRequest** 对象引用的每个组件 secret。
   a. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift Container Platform CLI。
   b. 获取所有引用的组件 secret 的名称和命名空间：

```
$ oc -n openshift-cloud-credential-operator get CredentialsRequest \
  -o json | jq -r '.items[] | select (.spec.providerSpec.kind=="<provider_spec>") |
.spec.secretRef'
```

   其中 **<provider_spec>** 是您的云供应商的对应值：

   - AWS: **AWSProviderSpec**
   - GCP: **GCPProviderSpec**

   AWS 输出的部分示例

```
{
  "name": "ebs-cloud-credentials",
  "namespace": "openshift-cluster-csi-drivers"
}
{
  "name": "cloud-credential-operator-iam-ro-creds",
  "namespace": "openshift-cloud-credential-operator"
}
```

   c. 删除每个引用的组件 secret：

```
$ oc delete secret <secret_name> \ 1
  -n <secret_namespace> 2
```

   1 指定 secret 的名称。

   2 指定包含 secret 的命名空间。

   删除 AWS secret 示例

```
$ oc delete secret ebs-cloud-credentials -n openshift-cluster-csi-drivers
```

您不需要从供应商控制台手动删除凭证。删除引用的组件 secret 将导致 CCO 从平台中删除现有凭证并创建新凭证。

验证

验证凭证是否已更改：

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Workloads → Secrets**。
2. 验证 **Value** 字段的内容已改变。

18.2.3.2. 删除云供应商凭证

在以 mint 模式使用 Cloud Credential Operator（CCO）安装 OpenShift Container Platform 集群后，您可以从集群中的 **kube-system** 命名空间中删除管理员级别的凭证 secret。只有在进行更改时（需要提高的权限，如升级），才需要管理员级别的凭证。

注意

在非 z-stream 升级前，您必须使用管理员级别的凭证重新恢复凭证 secret。如果没有凭证，则可能会阻止升级。

先决条件

- 集群安装在支持从 CCO 中删除云凭证的平台上。支持的平台是 AWS 和 GCP。

流程

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Workloads → Secrets**。
2. 在 **Secrets** 页面的表中，找到您的云供应商的 root secret。

平台	Secret 名称
AWS	**aws-creds**
GCP	**gcp-credentials**

3. 点击与 secret 相同的行  中的 **Options** 菜单，然后选择 **Delete Secret**。

18.2.4. 其他资源

- 用于 AWS 的 kube-system 项目中存储管理员级别的 secret 的替代方案
- GCP 的 kube-system 项目中存储管理员级别的 secret 的替代方案

18.3. 使用 PASSTHROUGH 模式

Amazon Web Services（AWS）、Microsoft Azure、Google Cloud Platform（GCP）、Red Hat OpenStack Platform（RHOSP）、Red Hat Virtualization（RHV）和 VMware vSphere 支持 passthrough 模式。

在 passthrough 模式中，Cloud Credential Operator（CCO）将提供的云凭证传递给请求云凭证的组件。凭证必须具有执行安装的权限，并可以完成集群中组件所需的操作，但并不需要可以创建新凭证的权限。CCO 不会尝试在 passthrough 模式中创建额外的有限范围凭证。

18.3.1. passthrough 模式权限要求

在使用 CCO 的 passthrough 模式时，请确保您提供的凭证满足运行或安装 OpenShift Container Platform 的云要求。如果提供的凭证由 CCO 传递给一个创建 **CredentialsRequest** CR 的组件，则该组件会在尝试调用没有权限的 API 时报告错误。

18.3.1.1. Amazon Web Services（AWS）权限

您在 AWS 中提供的 passthrough 模式的凭证必须具有您正在运行或安装的 OpenShift Container Platform 版本所需的所有 **credentialsRequest** CR 的所有请求权限。

要找到所需的 **CredentialsRequest** CR，请参阅为 AWS 手动创建 IAM。

18.3.1.2. Microsoft Azure 权限

您在 Azure 中提供的 passthrough 模式的凭证必须具有您正在运行或安装的 OpenShift Container Platform 版本所需的所有 **credentialsRequest** CR 的所有请求权限。

要找到所需的 **CredentialsRequest** CR，请参阅为 Azure 手动创建 IAM。

18.3.1.3. Google Cloud Platform（GCP）权限

您在 GCP 中提供的 passthrough 模式的凭证必须具有您正在运行或安装的 OpenShift Container Platform 版本所需的所有 **credentialsRequest** CR 的所有请求权限。

要找到所需的 **CredentialsRequest** CR，请参阅为 GCP 手动创建 IAM。

18.3.1.4. Red Hat OpenStack Platform（RHOSP）权限

要在 RHOSP 上安装 OpenShift Container Platform 集群，CCO 需要具有 **member** 用户角色权限的凭证。

18.3.1.5. Red Hat Virtualization（RHV）权限

要在 RHV 上安装 OpenShift Container Platform 集群，CCO 需要具有以下权限的凭证：

- **DiskOperator**
- **DiskCreator**
- **UserTemplateBasedVm**
- **TemplateOwner**
- **TemplateCreator**
- 在用于 OpenShift Container Platform 部署的特定集群中的 **ClusterAdmin**

18.3.1.6. VMware vSphere 权限

要在 VMware vSphere 上安装 OpenShift Container Platform 集群，CCO 需要具有以下 vSphere 权限的凭证：

表 18.2. 所需的 vSphere 权限

类别	权限
数据存储	*分配空间*
目录	*Create folder*、*Delete folder*
vSphere 标记	所有权限
网络	*分配网络*
资源	*为资源池分配虚拟机*
配置集驱动的存储	所有权限
vApp	所有权限
虚拟机器	所有权限

18.3.2. 管理凭证 root secret 格式

每个云供应商都使用 **kube-system** 命名空间中的一个凭证 root secret，用于满足所有凭证请求并创建它们对应的 secret。这可以通过 mint 新凭证使用 *mint 模式* 完成，或使用 *passthrough 模式* 复制凭证 root secret。

secret 的格式因云而异，也用于每个 **CredentialsRequest** secret。

Amazon Web Services(AWS)secret 格式

```
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 secret 格式

```
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>
```

在 Microsoft Azure 中，凭证 secret 格式包括两个必须包含集群基础架构 ID 的属性，每个集群安装随机生成。该值可在运行创建清单后找到：

```
$ cat .openshift_install_state.json | jq '."*installconfig.ClusterID".InfraID' -r
```

输出示例

```
mycluster-2mpcn
```

这个值将在 secret 数据中使用，如下所示：

```
azure_resource_prefix: mycluster-2mpcn
azure_resourcegroup: mycluster-2mpcn-rg
```

Google Cloud Platform(GCP)secret 格式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

Red Hat OpenStack Platform (RHOSP) secret 格式

```
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>
```

Red Hat Virtualization (RHV) secret 格式

```
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 secret 格式

```
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 模式凭证维护

如果 **CredentialsRequest** CR 随着集群升级而变化，您必须手动更新 passthrough 模式凭证以满足要求。为了避免升级过程中出现凭证问题，请在升级前检查发行镜像中的 **CredentialsRequest** CR。要找到云供应商所需的 **CredentialsRequest** CR,请参阅为 AWS、Azure 或 GCP *手动创建 IAM*。

18.3.3.1. 手动轮转云供应商凭证

如果因为某种原因更改了云供应商凭证，您必须手动更新 Cloud Credential Operator（CCO）用来管理云供应商凭证的 secret。

轮转云凭证的过程取决于 CCO 配置使用的模式。在为使用 mint 模式的集群轮转凭证后，您必须手动删除由删除凭证创建的组件凭证。

先决条件

- 您的集群会在支持使用您要使用的 CCO 模式手动轮转云凭证的平台上安装：
  - 对于 passthrough 模式，支持 Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Platform (GCP)、Red Hat OpenStack Platform (RHOSP)、Red Hat Virtualization (RHV)和 VMware vSphere。
- 您已更改了用于与云供应商接口的凭证。
- 新凭证有足够的权限来在集群中使用 CCO 模式。

流程

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Workloads → Secrets**。
2. 在 **Secrets** 页面的表中，找到您的云供应商的 root secret。

平台	Secret 名称
AWS	**aws-creds**
Azure	**azure-credentials**

平台	Secret 名称
GCP	**gcp-credentials**
RHOSP	**openstack-credentials**
RHV	**ovirt-credentials**
VMware vSphere	**vsphere-creds**

3. 点击与 secret 相同的行  中的 **Options** 菜单，然后选择 **Edit Secret**。
4. 记录 **Value** 字段的内容。您可以使用这些信息验证在更新凭证后该值是否不同。
5. 使用云供应商的新身份验证信息更新 **Value** 字段的文本，然后点 **Save**。
6. 如果您要为没有启用 vSphere CSI Driver Operator 的 vSphere 集群更新凭证，您必须强制推出 Kubernetes 控制器管理器以应用更新的凭证。

注意

如果启用了 vSphere CSI Driver Operator，则不需要这一步。

要应用更新的 vSphere 凭证，请以具有 **cluster-admin** 角色的用户身份登录到 OpenShift Container Platform CLI，并运行以下命令：

```
$ oc patch kubecontrollermanager cluster \
  -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date )"'"}}' \
  --type=merge
```

当凭证被推出时，Kubernetes Controller Manager Operator 的状态会报告 **Progressing=true**。要查看状态，请运行以下命令：

```
$ oc get co kube-controller-manager
```

验证

验证凭证是否已更改：

1. 在 web 控制台的 **Administrator** 视角中，导航到 **Workloads → Secrets**。
2. 验证 **Value** 字段的内容已改变。

其他资源

- vSphere CSI Driver Operator

18.3.4. 在安装后减少权限

在使用 passthrough 模式时，每个组件都有相同的权限供所有其他组件使用。如果您在安装后不减少权限，则所有组件都有运行安装程序所需的广泛权限。

安装后，您可以将凭证的权限减少到仅限运行集群所需的权限，这由您正在使用的 OpenShift Container Platform 版本的发行镜像中的 **CredentialsRequest** CR 定义。

要找到 AWS、Azure 或 GCP 所需的 **CredentialsRequest** CR，并了解如何更改 CCO 所用权限，请参阅为 AWS、Azure 或 GCP *手动创建 IAM*。

18.3.5. 其他资源

- 为 AWS 手动创建 IAM
- 为 Azure 手动创建 IAM
- 为 GCP 手动创建 IAM

18.4. 使用手动模式

Amazon Web Services（AWS）、Microsoft Azure 和 Google Cloud Platform（GCP）支持手动模式。

在手动模式中，用户管理云凭证而不是 Cloud Credential Operator（CCO）。要使用此模式，您必须检查发行镜像中用于运行或安装的 OpenShift Container Platform 版本的 **CredentialsRequest** CR，在底层云供应商中创建对应的凭证，并在正确的命名空间中创建 Kubernetes Secret，以满足集群云供应商的所有 **CredentialsRequest** CR。

使用手动模式可允许每个集群组件只拥有所需的权限，而无需在集群中存储管理员级别的凭证。此模式还需要连接到 AWS 公共 IAM 端点。但是，每次升级都必须手动将权限与新发行镜像协调。

有关将云供应商配置为使用手动模式的详情，请参阅为 AWS、Azure 或 GCP *手动创建 IAM*。

18.4.1. 使用 AWS STS 的手动模式

您可以使用手动模式配置 AWS 集群，以使用 Amazon Web Services Secure Token Service（AWS STS）。借助这一配置，CCO 对不同组件使用临时凭证。

18.4.2. 使用手动维护的凭证升级集群

默认情况下，带有手动维护凭证的集群的 Cloud Credential Operator(CCO)U **gradable 状态为 False**。

- 对于次发行版本（例如从 4.7 升级到 4.8），这个状态会阻止升级，直到您解决了任何更新的权限并 **添加了 CloudCredential** 资源，以指示下一版本根据需要更新权限。此注解将 **Upgradable** 状态更改为 **True**。
- 对于 z-stream 版本（例如从 4.8.9 到 4.8.10），不会添加或更改任何权限，因此不会阻止升级。

在使用手动维护凭证升级集群前，您必须为要升级到的发行镜像创建新凭证。另外，您必须检查现有凭证所需的权限，并满足新版本中这些组件的任何新权限要求。

流程

1. 提取并检查 **新版本的 CredentialsRequest** 自定义资源。
   详情请参阅您的云供应商的“手动创建 IAM”部分来了解如何获取和使用您的云所需的凭证。

2. 更新集群中手动维护的凭证：

   - 为新发行镜像添加的任何 **CredentialsRequest** 自定义资源创建新 secret。
   - 如果存储在 secret 中的任何现有凭证的 **CredentialsRequest** 自定义资源更改了其权限要求，请根据需要更新权限。

3. 当所有 secret 都对新发行版本正确时，表示集群已准备好升级：

   a. 以具有 **cluster-admin** 角色的用户身份登录 OpenShift Container Platform CLI。

   b. 编辑 **CloudCredential** 资源，以在 **metadata** 字段中添加 **可升级至** 注解：

   ```
   $ oc edit cloudcredential cluster
   ```

   要添加的文本

   ```
   ...
     metadata:
       annotations:
         cloudcredential.openshift.io/upgradeable-to: <version_number>
   ...
   ```

   其中 **<version_number>** 是您要升级到的版本，格式为 **x.y.z**。例如，OpenShift Container Platform **4.8.2** 代表 OpenShift Container Platform 4.8.2。

   添加可升级状态进行更改的注解后，可能需要几分钟时间。

4. 验证 CCO 是否可升级：

   a. 在 Web 控制台的 **Administrator 视角中**，导航到 **Administration → Cluster Settings**。

   b. 要查看 CCO 状态详情，请点击 **Cluster Operators** 列表中的 **cloud-credential**。

   c. 如果 **Conditions** 部分中的 **Upgradeable** 状态为 **False**，请验证 **upgradeable-to** 注解没有拼写错误。

当 **Conditions** 部分中的 **Upgradeable** 状态为 **True** 时，您可以开始 OpenShift Container Platform 升级。

18.4.3. 其他资源

- 为 AWS 手动创建 IAM
- 为 Azure 手动创建 IAM
- 为 GCP 手动创建 IAM
- 使用 AWS STS 的手动模式

18.5. 在 AMAZON WEB SERVICES SECURE TOKEN SERVICE 中使用手动模式

Amazon Web Services（AWS）支持使用 STS 的手动模式。

注意

此凭证策略只支持新的 OpenShift Container Platform 集群，且必须在安装过程中进行配置。您无法重新配置使用不同凭证策略的现有集群，以使用此功能。

18.5.1. 关于 AWS Secure Token Service 的手动模式

在带有 STS 的手动模式中，各个 OpenShift Container Platform 集群组件使用 AWS Secure Token Service（STS）来分配提供简短、有限权限安全凭证的组件 IAM 角色。这些凭证与特定于发布 AWS API 调用的每个组件的 IAM 角色关联。

使用适当配置的 AWS IAM OpenID Connect（OIDC）身份提供程序以及 AWS IAM 角色会自动请求新的和刷新的凭证。OpenShift Container Platform 为服务帐户令牌签名，这些令牌由 AWS IAM 信任，并可投射到 pod 中并用于身份验证。令牌会在一小时后刷新。

图 18.1. STS 验证流程

使用带有 STS 的手动模式更改为各个 OpenShift Container Platform 组件提供的 AWS 凭证的内容。

使用长期凭证的 AWS secret 格式

```
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 组件的命名空间。

2 组件 secret 的名称。

使用 STS 的 AWS secret 格式

```
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. 组件的命名空间。
2. 组件 secret 的名称。
3. 组件的 IAM 角色。
4. pod 中服务帐户令牌的路径。通常这是 OpenShift Container Platform 组件的 **/var/run/secrets/openshift/serviceaccount/token**。

18.5.2. 使用 STS 为手动模式安装 OpenShift Container Platform 集群

要安装配置为在带有 STS 的手动模式中使用 Cloud Credential Operator（CCO）的集群：

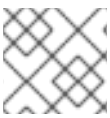
1. 配置 Cloud Credential Operator 工具。
2. 单独创建所需的 AWS 资源，或使用一个命令创建。
3. 运行 OpenShift Container Platform 安装程序
4. 验证集群是否使用简短提供的凭证。

注意

因为在使用 STS 时集群以手动模式运行，所以无法使用所需的权限为组件创建新凭证。当升级到不同版本的 OpenShift Container Platform 时，通常会有新的 AWS 权限要求。在升级使用 STS 的集群前，集群管理员必须手动确保 AWS 权限足以用于现有组件，并可供任何新组件使用。

18.5.2.1. 配置 Cloud Credential Operator 工具

当 Cloud Credential Operator（CCO）使用 STS 以手动模式运行时，要从集群外创建和管理云凭证，提取并准备 CCO 实用程序（**ccoctl**）二进制文件。



注意

ccoctl 是一个必须在 Linux 环境中运行的 Linux 二进制文件。

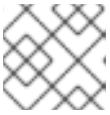
流程

1. 获取 OpenShift Container Platform 发行镜像。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 从 OpenShift Container Platform 发行镜像获取 CCO 容器镜像：

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE)
```



注意

确保 **$RELEASE_IMAGE** 的架构与将使用 **ccoctl** 工具的环境架构相匹配。

3. 将 CCO 容器镜像中的 **ccoctl** 二进制文件提取到 OpenShift Container Platform 发行镜像中：

```
$ oc image extract $CCO_IMAGE --file="/usr/bin/ccoctl" -a ~/.pull-secret
```

4. 更改权限以使 **ccoctl** 可执行：

```
$ chmod 775 ccoctl
```

验证

- 要验证 **ccoctl** 是否可以使用，请显示帮助文件：

```
$ ccoctl aws --help
```

ccoctl aws --help 的输出：

```
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.
```

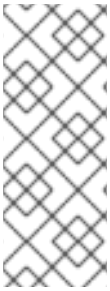
18.5.2.2. 使用 Cloud Credential Operator 实用程序创建 AWS 资源

您可以使用 CCO 实用程序（**ccoctl**）单独创建所需的 AWS 资源，或使用单个命令来创建。

18.5.2.2.1. 单独创建 AWS 资源

如果您需要在修改 AWS 资源前查看 **ccoctl** 工具创建的 JSON 文件，或者 **ccoctl** 工具用于创建 AWS 资源的过程无法自动满足组织的要求，您可以单独创建 AWS 资源。例如，此选项对于在不同用户或部门之间共享创建这些资源的组织可能有用。

否则，您可以使用 **ccoctl aws create-all** 命令自动创建 AWS 资源。



注意

默认情况下，**ccoctl** 在运行命令的目录中创建对象。要在其他目录中创建对象，请使用 **--output-dir** 标志。此流程使用 **<path_to_ccoctl_output_dir>** 来引用这个目录。

有些 **ccoctl** 命令会发出 AWS API 调用来创建或修改 AWS 资源。您可以使用 **--dry-run** 标志来避免 API 调用。使用此标志可在本地文件系统中创建 JSON 文件。您可以使用 **--cli-input-json** 参数查看和修改 JSON 文件，然后使用 AWS CLI 工具应用它们。

先决条件

- 提取并准备 **ccoctl** 二进制文件。

流程

1. 生成用于为集群设置 OpenID Connect 供应商的公共和私有 RSA 密钥文件：

```
$ ccoctl aws create-key-pair
```

输**出示例：**

```
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
```

其中 **serviceaccount-signer.private** 和 **serviceaccount-signer.public** 是生成的密钥文件。

此命令还会在 **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** 中创建集群在安装过程中所需的私钥。

2. 在 AWS 上创建 OpenID Connect 身份提供程序和 S3 存储桶：

```
$ ccoctl aws create-identity-provider \
--name=<name> \
--region=<aws_region> \
--public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public
```

其中：

- **<name>** 是用于标记为跟踪而创建的云资源的名称。
- **<aws-region>** 是将要在其中创建云资源的 AWS 区域。
- **<path_to_ccoctl_output_dir>** 是 **ccoctl aws create-key-pair** 命令生成的公钥文件的路径。

输**出示例：**

```
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
```

其中 **02-openid-configuration** 是发现文档，**03-keys.json** 是 JSON Web 密钥集文件。

此命令还会在 **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** 中创建 YAML 配置文件。此文件为集群生成的服务帐户令牌设置签发者 URL 字段，以便 AWS IAM 身份提供程序信任令牌。

3. 为集群中的每个组件创建 IAM 角色。

   a. 从 OpenShift Container Platform 发行镜像中提取 **CredentialsRequest** 对象列表：

```
$ 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** 是存储 **CredentialsRequest** 对象列表的目录。如果该目录不存在，此命令就会创建该目录。

   b. 使用 **ccoctl** 工具处理 **credrequests** 目录中的所有 **CredentialsRequest** 对象：

```
$ 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
```

注意

对于使用其他 IAM API 端点的 AWS 环境（如 GovCloud），还必须使用 **--region** 参数指定您的区域。

对于每个 **CredentialsRequest** 对象，**ccoctl** 创建一个带有信任策略的 IAM 角色，该角色与指定的 OIDC 身份提供程序相关联，以及来自 OpenShift Container Platform 发行镜像的每个 **CredentialsRequest** 对象中定义的权限策略。

验证

- 要验证 OpenShift Container Platform secret 是否已创建，列出 **<path_to_ccoctl_output_dir>/manifests** 目录中的文件：

```
$ ll <path_to_ccoctl_output_dir>/manifests
```

输出示例：

```
total 24
```

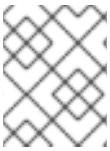
```
-rw-------. 1 <user> <user> 161 Apr 13 11:42 cluster-authentication-02-config.yaml
-rw-------. 1 <user> <user> 379 Apr 13 11:59 openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
-rw-------. 1 <user> <user> 353 Apr 13 11:59 openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
-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
```

您可以通过查询 AWS 来验证是否已创建 IAM 角色。如需更多信息，请参阅有关列出 IAM 角色的 AWS 文档。

18.5.2.2.2. 使用单个命令创建 AWS 资源

如果您不需要在修改 AWS 资源前查看 **ccoctl** 工具创建的 JSON 文件，并且如果 **ccoctl** 工具用于创建 AWS 资源的过程会自动满足您的要求，则可以使用 **ccoctl aws create-all** 命令自动创建 AWS 资源。

否则，您可以单独创建 AWS 资源。



注意

默认情况下，**ccoctl** 在运行命令的目录中创建对象。要在其他目录中创建对象，请使用 **--output-dir** 标志。此流程使用 **<path_to_ccoctl_output_dir>** 来引用这个目录。

先决条件

- 提取并准备 **ccoctl** 二进制文件。

流程

1. 从 OpenShift Container Platform 发行镜像中提取 **CredentialsRequest** 对象列表：

```
$ 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** 是存储 **CredentialsRequest** 对象列表的目录。如果该目录不存在，此命令就会创建该目录。

2. 使用 **ccoctl** 工具处理 **credrequests** 目录中的所有 **CredentialsRequest** 对象：

```
$ ccoctl aws create-all \
--name=<name> \
--region=<aws_region> \
--credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests
```

验证

- 要验证 OpenShift Container Platform secret 是否已创建，列出 **<path_to_ccoctl_output_dir>/manifests** 目录中的文件：

```
$ ll <path_to_ccoctl_output_dir>/manifests
```

输出示例：

```
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
```

您可以通过查询 AWS 来验证是否已创建 IAM 角色。如需更多信息，请参阅有关列出 IAM 角色的 AWS 文档。

18.5.2.3. 运行安装程序

先决条件

- 获取 OpenShift Container Platform 发行镜像。

流程

1. 进入包含安装程序的目录并创建 **install-config.yaml** 文件：

```
$ openshift-install create install-config --dir <installation_directory>
```

其中 **<installation_directory>** 是安装程序在其中创建文件的目录。

2. 编辑 **install-config.yaml** 配置文件，使其包含将 **credentialsMode** 参数设置为 **Manual**。

install-config.yaml 配置文件示例

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
...
```

(1) 添加这一行将 **credentialsMode** 参数设置为 **Manual**。

3. 创建所需的 OpenShift Container Platform 安装清单：

```
$ openshift-install create manifests
```

4. 将 **ccoctl** 生成的清单复制到安装程序创建的 manifests 目录中：

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. 将 **ccoctl** 在 **tls** 目录中生成的私钥复制到安装目录中：

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6. 运行 OpenShift Container Platform 安装程序：

```
$ ./openshift-install create cluster
```

18.5.2.4. 验证安装

1. 连接到 OpenShift Container Platform 集群。
2. 验证集群没有 **root** 凭证：

```
$ oc get secrets -n kube-system aws-creds
```

输出应类似于：

```
Error from server (NotFound): secrets "aws-creds" not found
```

3. 验证组件是否假定 secret 清单中指定的 IAM 角色，而不是使用由 CCO 创建的凭证：

带有 Image Registry Operator 的命令示例

```
$ oc get secrets -n openshift-image-registry installer-cloud-credentials -o json | jq -r
.data.credentials | base64 --decode
```

输出应显示组件使用的角色和 Web 身份令牌，如下所示：

带有 Image Registry Operator 的输出示例

```
[default]
role_arn = arn:aws:iam::123456789:role/openshift-image-registry-installer-cloud-credentials
web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
```