Red Hat OpenShift Service on AWS 4

认证和授权

Red Hat OpenShift Service on AWS 集群的安全。

Last Updated: 2024-07-03

Red Hat OpenShift Service on AWS 4 认证和授权

Red Hat OpenShift Service on AWS 集群的安全。

法律通告

摘要

本文档提供有关升级 Red Hat OpenShift Service on AWS 集群的安全信息。

目录

第 1 章 身份验证和授权概述

1.1. RED HAT OPENSHIFT SERVICE ON AWS 身份验证和授权的常见术语表

该术语表定义了 Red Hat OpenShift Service on AWS 身份验证和授权中使用的常用术语。

身份验证

身份验证决定了访问 Red Hat OpenShift Service on AWS 集群，并确保只有经过身份验证的用户可以访问 Red Hat OpenShift Service on AWS 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。

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

由软件及其所有依赖项组成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、公有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。

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 Red Hat OpenShift Service on AWS 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

Red Hat OpenShift Service on AWS control plane 包括一个内置的 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。

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。

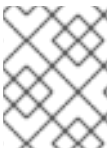
users

用户是可以向 API 发出请求的实体。

1.2. 关于 RED HAT OPENSHIFT SERVICE ON AWS 中的身份验证

为了控制对 Red Hat OpenShift Service on AWS 集群上的 Red Hat OpenShift Service 的访问，具有 **dedicated-admin** 角色的管理员可以配置 用户身份验证，并确保只有批准的用户访问集群。

要与 Red Hat OpenShift Service on AWS 集群交互，用户必须首先以某种方式向 Red Hat OpenShift Service on AWS API 进行身份验证。您可以通过在对 Red Hat OpenShift Service on AWS 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。

管理员可以通过配置身份提供程序来配置身份验证。您可以在 Red Hat OpenShift Service on AWS 中定义任何支持的身份提供程序，并将其添加到集群中。

1.3. 关于 RED HAT OPENSHIFT SERVICE ON AWS 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。要了解授权如何在 AWS 上的 Red Hat OpenShift Service 中工作，请参阅 评估授权。

您还可以通过项目和命名空间来控制对 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 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。

- 为用户或组分配集群角色： Red Hat OpenShift Service on AWS 包括一组默认集群角色。您可以将 他们添加到用户或组 中。
- 创建 cluster-admin 和 dedicated-admin 用户：在 AWS 集群上创建 Red Hat OpenShift Service 的用户可以授予其他 **cluster-admin** 和 **dedicated-admin** 用户的访问权限。
- 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。用户可以在应用程序中创建并使用一个服务账户，也可以作为一个 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 Red Hat OpenShift Service on AWS 用户组同步，在一个位置管理用户组。

第 2 章 了解身份验证

要使用户与 Red Hat OpenShift Service on AWS 交互，必须首先对集群进行身份验证。身份验证层标识了与对 Red Hat OpenShift Service on AWS API 的请求关联的用户。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。

2.1. 用户

Red Hat OpenShift Service on AWS 中的用户是可以向 Red Hat OpenShift Service on AWS API 发出请求的实体。*Red Hat OpenShift Service on AWS* ***User*** *对象代表一个操作者，它可以通过向它们或其组添加角色来授予系统中的权限。通常，这代表与 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 用户的方式。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**

*每个用户都必须通过某种方式进行身份验证才能访问 Red Hat OpenShift Service on AWS。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 Red Hat OpenShift Service on AWS 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 AWS OAuth 服务器上的 Red Hat OpenShift Service 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. Red Hat OpenShift Service on AWS OAuth 服务器

Red Hat OpenShift Service on AWS master 包括内置的 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 Red Hat OpenShift Service on AWS 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。大多数身份验证集成都会在这个端点前面放置身份验证代理，或配置 Red Hat OpenShift Service on AWS，以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，除了交互式登录流外，Red Hat OpenShift Service on AWS 支持使用 **WWW-Authenticate** 质询进行身份验证。*

*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 Red Hat OpenShift Service on AWS 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。*

第 3 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。

3.1. 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
```

3.2. 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。

3.3. 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
```

3.4. 将未经身份验证的组添加到集群角色中

作为集群管理员，您可以通过创建集群角色绑定，将未经身份验证的用户添加到 Red Hat OpenShift Service on AWS 中的以下集群角色中。未经身份验证的用户无法访问非公共集群角色。这只应在需要时在特定用例中完成。

您可以将未经身份验证的用户添加到以下集群角色中：

- **system:scope-impersonation**
- **system:webhook**
- **system:oauth-token-deleter**
- **self-access-reviewer**

重要

在修改未经身份验证的访问时，始终验证符合您机构的安全标准。

先决条件

- 您可以使用具有 **cluster-admin** 角色的用户访问集群。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 创建名为 **add-<cluster_role>-unauth.yaml** 的 YAML 文件，并添加以下内容：

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 annotations:
   rbac.authorization.kubernetes.io/autoupdate: "true"
 name: <cluster_role>access-unauthenticated
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: <cluster_role>
subjects:
 - apiGroup: rbac.authorization.k8s.io
   kind: Group
   name: system:unauthenticated
```

2. 运行以下命令来应用配置：

```
$ oc apply -f add-<cluster_role>.yaml
```

第 4 章 配置身份提供程序

创建 Red Hat OpenShift Service on AWS 集群后，您必须配置身份提供程序，以确定用户如何登录以访问集群。

*以下主题描述了如何使用 OpenShift Cluster Manager 控制台配置身份提供程序。另外，您可以使用 ROSA CLI (**rosa**) 来配置身份提供程序并访问集群。*

4.1. 了解身份提供程序

Red Hat OpenShift Service on AWS 包括内置的 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。配置身份提供程序可让用户登录和访问集群。

4.1.1. 支持的身份提供程序

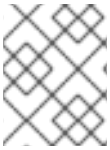
您可以配置以下类型的身份提供程序：

用户身份提供程序	描述
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，以使用 GitLab.com 或任何其他 GitLab 实例作为身份提供程序。
Google	使用 Google's OpenID Connect integration 配置 Google 身份提供程序。
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。
OpenID Connect	配置 OpenID Connect (OIDC) 身份提供程序，以使用授权代码流与 OIDC 身份提供程序集成。
htpasswd	为单个静态管理用户配置 htpasswd 身份提供程序。您可以以用户身份登录到集群来排除问题。  **重要** htpasswd 身份提供程序选项仅用于创建单个静态管理用户。htpasswd 不支持作为 Red Hat OpenShift Service on AWS 的通用身份提供程序。有关配置单个用户的步骤，请参阅*配置 htpasswd 身份提供程序*。

4.1.2. 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。
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。 **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。*

4.2. 配置 GITHUB 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，并访问 Red Hat OpenShift Service on AWS 集群。OAuth 有助于 Red Hat OpenShift Service on AWS 和 GitHub 或 GitHub Enterprise 之间的令牌交换流。

警告

配置 GitHub 身份验证后，用户可以使用 GitHub 凭证登录到 Red Hat OpenShift Service on AWS。要防止具有任何 GitHub 用户 ID 的任何人登录到 Red Hat OpenShift Service on AWS 集群，您必须只限制对特定 GitHub 机构或团队中的访问。

先决条件

- *OAuth 应用程序必须直接由 GitHub 机构管理员在 GitHub 机构设置中创建。*
- *GitHub 机构或团队 在您的 GitHub 帐户中设置。*

流程

1. *在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面，再选择您需要为其配置身份提供程序的集群。*
2. *点 **Access control** 选项卡。*

3. 点 **Add identity provider**。

注意

您还可以点在集群创建后显示的警告信息中的 **Add Oauth 配置** 链接来配置身份提供程序。

4. 从下拉菜单中选择 **GitHub**。
5. 输入身份提供程序的唯一名称。之后无法更改此名称。
   - 在提供的字段中自动生成 **OAuth 回调 URL**。您将使用它来注册 GitHub 应用。

     ```
     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. 在 GitHub 上注册应用程序。
7. 返回到 Red Hat OpenShift Service on AWS，然后从下拉菜单中选择映射方法。在大多数情况下推荐使用 **声明**。
8. 输入 GitHub 提供的**客户端 ID** 和**客户端 secret**。
9. 输入一个**主机名**。在使用托管 GitHub Enterprise 实例时，必须输入一个主机名。
10. 可选：您可以指定证书颁发机构 (CA) 文件来验证配置的 GitHub Enterprise URL 的服务器证书。点 **Browse** 找到并附加 **CA 文件**到身份提供程序。
11. 选择 **Use organizations** 或 **Use teams** 以限制对特定 GitHub 组织或 GitHub 团队的访问。
12. 输入您要限制访问权限的机构或团队的名称。点 **Add more** 指定用户可以成为用户所属的多个机构或团队。
13. 单击 **Confirm**。

验证

- 配置的身份提供程序可以在 **Cluster details** 页面的 **Access control** 选项卡中看到。

4.3. 配置 GITLAB 身份提供程序

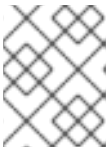
配置 GitLab 身份提供程序，以使用 GitLab.com 或任何其他 GitLab 实例作为身份提供程序。

先决条件

- 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。

流程

1. 在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面，再选择您需要为其配置身份提供程序的集群。
2. 点 **Access control** 选项卡。
3. 点 **Add identity provider**。



注意

您还可以点在集群创建后显示的警告信息中的 **Add Oauth 配置** 链接来配置身份提供程序。

4. 从下拉菜单中选择 **GitLab**。
5. 输入身份提供程序的唯一名称。之后无法更改此名称。
   - 在提供的字段中自动生成 **OAuth 回调 URL**。您将提供此 URL 到 GitLab。

     ```
     https://oauth-openshift.apps.<cluster_name>.
     <cluster_domain>/oauth2callback/<idp_provider_name>
     ```

     例如：

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

6. 在 GitLab 中添加新应用程序。
7. 返回到 Red Hat OpenShift Service on AWS，然后从下拉菜单中选择映射方法。在大多数情况下推荐使用 **声明**。
8. 输入 GitLab 提供的**客户端 ID** 和**客户端 secret**。
9. 输入 GitLab 供应商的 **URL**。
10. 可选：您可以使用证书颁发机构 (CA) 文件来验证配置的 GitLab URL 的服务器证书。点 **Browse** 找到并附加 **CA 文件**到身份提供程序。
11. 单击 **Confirm**。

验证

- 配置的身份提供程序可以在 **Cluster details** 页面的 **Access control** 选项卡中看到。

4.4. 配置 GOOGLE 身份提供程序

配置 Google 身份提供程序，以便用户通过 Google 凭证进行身份验证。

警告

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。

流程

1. 在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面，再选择您需要为其配置身份提供程序的集群。
2. 点 **Access control** 选项卡。
3. 点 **Add identity provider**。

注意

您还可以点在集群创建后显示的警告信息中的 **Add Oauth 配置** 链接来配置身份提供程序。

4. 从下拉菜单中选择 **Google**。
5. 输入身份提供程序的唯一名称。之后无法更改此名称。
   - 在提供的字段中自动生成 **OAuth 回调 URL**。您将为 Google 提供此 URL。

     ```
     https://oauth-openshift.apps.<cluster_name>.
     <cluster_domain>/oauth2callback/<idp_provider_name>
     ```

     例如：

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

6. 使用 Google's OpenID Connect integration 配置 Google 身份提供程序。
7. 返回到 Red Hat OpenShift Service on AWS，然后从下拉菜单中选择映射方法。在大多数情况下推荐使用 **声明**。
8. 输入注册 Google 项目的**客户端 ID**，以及 Google 发布的**客户端 secret**。
9. 输入**托管域**，将用户限制到 Google Apps 域。
10. 单击 **Confirm**。

验证

- 配置的身份提供程序可以在 **Cluster details** 页面的 **Access control** 选项卡中看到。

4.5. 配置 LDAP 身份提供程序

配置 LDAP 身份提供程序，以使用简单绑定身份验证针对 LDAPv3 服务器验证用户名和密码。

先决条件

- *在配置 LDAP 身份提供程序时，您需要输入配置的 **LDAP URL**。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*

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>))*

重要

*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**。*

流程

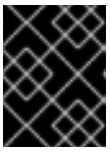
1. *在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面，再选择您需要为其配置身份提供程序的集群。*
2. *点 **Access control** 选项卡。*
3. *点 **Add identity provider**。*

注意

*您还可以点在集群创建后显示的警告信息中的 **Add Oauth 配置** 链接来配置身份提供程序。*

4. *从下拉菜单中选择 **LDAP**。*

5. 输入身份提供程序的唯一名称。之后无法更改此名称。
6. 从下拉菜单中选择映射方法。在大多数情况下推荐使用 **声明**。
7. 输入 **LDAP URL** 以指定要使用的 LDAP 搜索参数。
8. 可选：输入 **绑定 DN** 和 **绑定密码**。
9. 输入将 LDAP 属性映射到身份的属性。
   - 输入 **ID** 属性，其值应用作用户 ID。点 **Add more** 来添加多个 ID 属性。
   - 可选：输入一个 **Preferred username** 属性，其值应用作显示名称。点 **Add more** 来添加多个首选用户名属性。
   - 可选：输入 **Email** 属性，其值应用作电子邮件地址。点 **Add more** 来添加多个电子邮件属性。
10. 可选：点 **Show advanced Options** 将证书颁发机构 (CA) 文件添加到 LDAP 身份提供程序中，以验证所配置 URL 的服务器证书。点 **Browse** 找到并附加 **CA 文件**到身份提供程序。
11. 可选：在高级选项下，您可以选择使 LDAP 供应商**不安全**。如果您选择这个选项，则无法使用 CA 文件。



重要

如果您使用不安全的 LDAP 连接（ldap:// 或端口 389），则必须在配置向导中检查 **Insecure** 选项。

12. 单击 **Confirm**。

验证

- 配置的身份提供程序可以在 **Cluster details** 页面的 **Access control** 选项卡中看到。

4.6. 配置 OPENID 身份提供程序

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

重要

Red Hat OpenShift Service on AWS 中的 Authentication Operator 要求配置的 OpenID Connect 身份提供程序实现 OpenID Connect Discovery 规格。

声明可读取自从 OpenID 身份提供程序返回的 JWT **id_token**；若有指定，也可读取自从 Issuer URL 返回的 JSON。

必须至少配置一个声明，以用作用户的身份。

您还可以指定将哪些声明用作用户的首选用户名、显示名称和电子邮件地址。如果指定了多个声明，则使用第一个带有非空值的声明。标准的声明是：

声明	描述
preferred_username	置备用户时的首选用户名。用户希望使用的简写名称，如 **janedoe**。通常，与身份验证系统中用户的登录或用户名对应的值，如用户名或电子邮件。
email	电子邮件地址。
name	显示名称。

如需更多信息，请参阅 OpenID 声明文档。

先决条件

- *在配置 OpenID Connect 前，请查看您要用于 Red Hat OpenShift Service on AWS 集群的任何红帽产品或服务的安装先决条件。*

流程

1. *在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面，再选择您需要为其配置身份提供程序的集群。*
2. *点 **Access control** 选项卡。*
3. *点 **Add identity provider**。*

注意

*您还可以点在集群创建后显示的警告信息中的 **Add Oauth 配置** 链接来配置身份提供程序。*

4. *从下拉菜单中选择 **OpenID**。*
5. *输入身份提供程序的唯一名称。之后无法更改此名称。*
   - *在提供的字段中自动生成 **OAuth 回调 URL**。*

     > *https://oauth-openshift.apps.<cluster_name>.<cluster_domain>/oauth2callback/<idp_provider_name>*

     例如：

     > *https://oauth-openshift.apps.openshift-cluster.example.com/oauth2callback/openid*

6. *按照创建授权请求中的步骤，在 OpenID 身份提供程序中注册新的 OpenID Connect 客户端。*
7. *返回到 Red Hat OpenShift Service on AWS，然后从下拉菜单中选择映射方法。在大多数情况下推荐使用 **声明**。*
8. *输入 OpenID 提供的**客户端 ID** 和**客户端 secret**。*

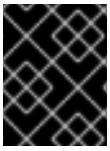
9. 输入 **Issuer URL**。这是 OpenID 供应商断言为 Issuer 标识符的 URL。它必须使用没有 URL 查询参数或片段的 https 方案。
10. 输入 **Email** 属性，其值应用作电子邮件地址。点 **Add more** 来添加多个电子邮件属性。
11. 输入 **Name** 属性，其值应用作首选用户名。点 **Add more** 来添加多个首选用户名。
12. 输入 **Preferred username** 属性，其值应用作显示名称。点 **Add more** 来添加多个显示名称。
13. 可选：点 **Show advanced Options** 将证书颁发机构 (CA) 文件添加到 OpenID 身份提供程序中。
14. 可选：在高级选项下，您可以添加 **其他范围**。默认情况下，请求 **OpenID** 范围。
15. 单击 **Confirm**。

验证

- 配置的身份提供程序可以在 **Cluster details** 页面的 **Access control** 选项卡中看到。

4.7. 配置 HTPASSWD 身份提供程序

配置 htpasswd 身份提供程序，以创建具有集群管理特权的单个静态用户。您可以以用户身份登录集群来排除问题。

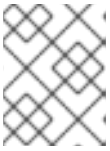


重要

htpasswd 身份提供程序选项仅用于创建单个静态管理用户。htpasswd 不支持作为 Red Hat OpenShift Service on AWS 的通用身份提供程序。

流程

1. 在 OpenShift Cluster Manager 中，进入到 **Clusters** 页面并选择您的集群。
2. 选择 **Access control** → **Identity provider**。
3. 点 **Add identity provider**。
4. 从 **Identity Provider** 下拉菜单中选择 **HTPasswd**。
5. 在身份提供程序的 **Name** 字段中添加唯一名称。
6. 为静态用户使用推荐的用户名和密码，或者自行创建。



注意

在以下步骤中选择 **Add** 后，此步骤中定义的凭证不可见。如果丢失了凭证，您必须重新创建身份提供程序并再次定义凭证。

7. 选择 **Add** 来创建 htpasswd 身份提供程序和单一静态用户。
8. 授予静态用户权限来管理集群：
   a. 在 **Access control** → **Cluster Roles and Access** 下，选择 **Add user**。
   b. 输入您在上一步中创建的静态用户的**用户 ID**。

c. 选择 **Add user** 为用户授予管理权限。

验证

- 配置的 htpasswd 身份提供程序在 **Access control → Identity provider** 页面中可见。

注意

创建身份提供程序后，同步通常在两分钟内完成。您可以在 htpasswd 身份提供程序可用后以用户身份登录集群。

- 单、管理用户在 **Access control → Cluster Roles 和 Access** 页面中可见。也会显示用户的管理组成员资格。

4.8. 其他资源

- 访问集群
- 了解使用 STS 部署工作流的 ROSA

第 5 章 使用 RBAC 定义和应用权限

5.1. 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。

具有 **dedicated-admin** 角色的管理员可以利用集群角色和绑定来控制谁对 Red Hat OpenShift Service on AWS 平台本身和所有项目都有各种访问权限级别。

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。

5.1.1. 默认集群角色

Red Hat OpenShift Service on AWS 包括了一组默认集群角色，您可以在集群范围或本地绑定到用户和组。

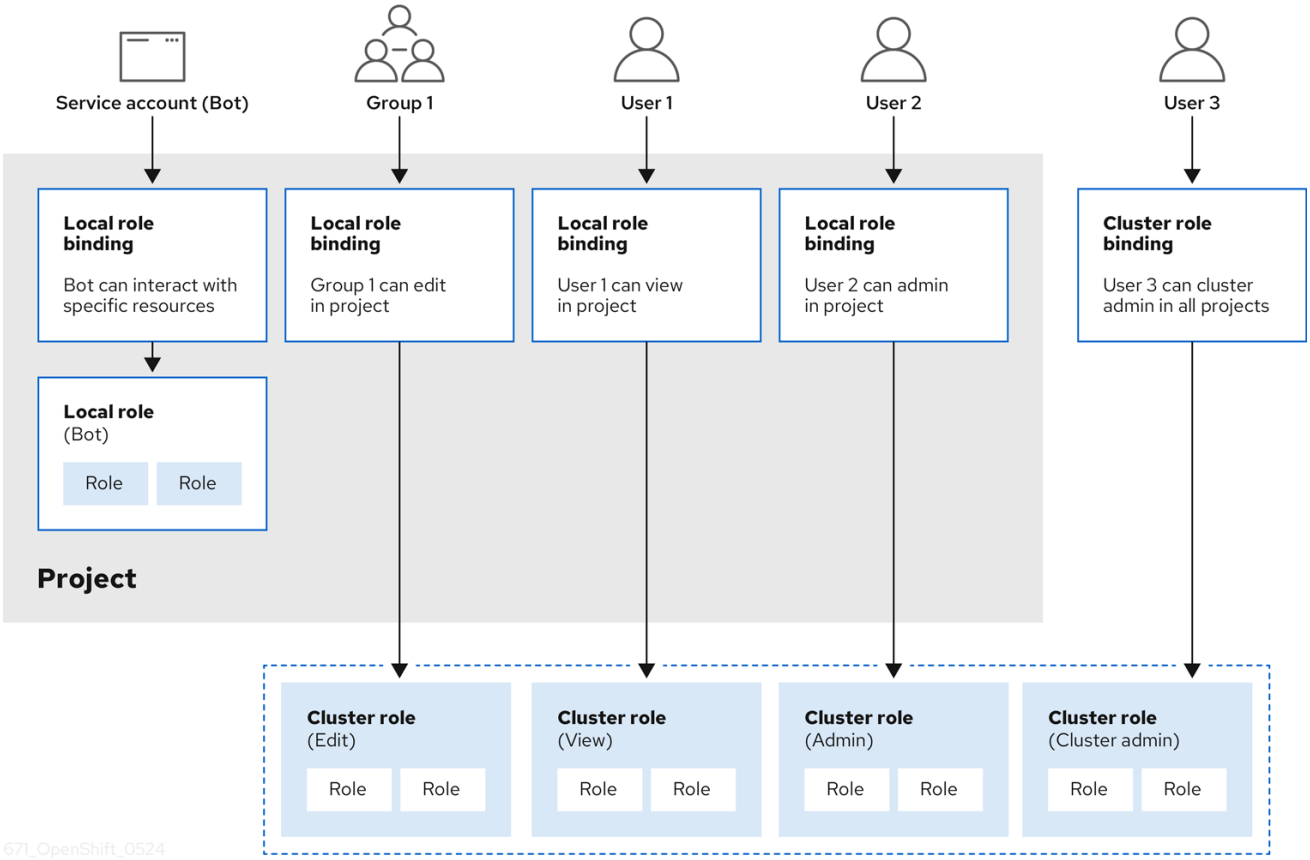
重要

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。



警告

*当它们被应用到一个角色时，**get pods/exec**, **get pods/***, 和 **get *** 规则会授予执行权限。应用最小特权的原则，仅分配用户和代理所需的最小 RBAC 权限。如需更多信息，请参阅 RBAC 规则允许执行权限。*

5.1.2. 评估授权

Red Hat OpenShift Service on AWS 通过以下命令评估授权：

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。

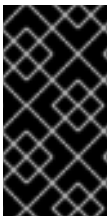
Red Hat OpenShift Service on AWS 通过以下步骤评估授权：

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。

5.1.2.1. 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。

5.2. 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：

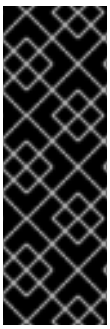
对象	描述
对象（object）	Pod、服务和复制控制器等。
策略（policy）	用户能否对对象执行操作的规则。
约束（constraint）	对各种对象进行限制的配额。
服务帐户	服务帐户自动使用项目中对象的指定访问权限进行操作。

具有 **dedicated-admin** 角色的管理员可以创建项目，并将项目的管理权限委派给用户社区的任何成员。具有 **dedicated-admin** 角色的管理员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。

5.3. 默认项目

Red Hat OpenShift Service on AWS 附带多个默认项目，以 **openshift-** 开头的项目对用户来说是最重要的。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"。



重要

不要在默认项目中运行工作负载或共享对默认项目的访问权限。为运行核心集群组件保留默认项目。

以下默认项目被视为具有高度特权：**default**, **kube-public**, **kube-system**, **openshift**, **openshift-infra**, **openshift-node**，其他系统创建的项目的标签 **openshift.io/run-level** 被设置为 **0** 或 **1**。依赖于准入插件（如 pod 安全准入、安全性上下文约束、集群资源配额和镜像引用解析）的功能无法在高特权项目中工作。

5.4. 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。

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

...
```

5.5. 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：
  - 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
```

5.6. 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**。*

5.7. 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
```

5.8. 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 5.1. 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。

命令	描述
$ 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。

5.9. 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 5.2. 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。

5.10. 授予 CLUSTER-ADMIN 访问权限

*以创建集群的用户身份，将 **cluster-admin** 用户角色添加到您的帐户中，使其具有最大管理员特权。创建集群时，这些权限不会自动分配给您的用户帐户。*

*另外，只有创建集群的用户才能向其他 **cluster-admin** 或 **dedicated-admin** 用户授予集群访问权限。具有 **dedicated-admin** 访问权限的用户具有较少的特权。作为最佳实践，将 **cluster-admin** 用户数量限制为尽量少。*

先决条件

- *您已在集群中添加身份提供程序(IDP)。*
- *您有要创建的用户的 IDP 用户名。*
- *已登陆到集群。*

流程

1. *授予用户 **cluster-admin** 权限：*

```
$ rosa grant user cluster-admin --user=<idp_user_name> --cluster=<cluster_name>
```

2. 验证您的用户是否被列为集群管理员：

```
$ rosa list users --cluster=<cluster_name>
```

输出示例

```
GROUP            NAME
cluster-admins   rh-rosa-test-user
dedicated-admins rh-rosa-test-user
```

3. 输入以下命令验证您的用户现在是否具有 **cluster-admin** 访问权限。集群管理员可以在不出错的情况下运行此命令，但专用管理员无法运行。

```
$ oc get all -n openshift-apiserver
```

输出示例

```
NAME                 READY   STATUS    RESTARTS   AGE
pod/apiserver-6ndg2  1/1     Running   0          17h
pod/apiserver-lrmxs  1/1     Running   0          17h
pod/apiserver-tsqhz  1/1     Running   0          17h
NAME          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/api   ClusterIP   172.30.23.241  <none>        443/TCP   18h
NAME                    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR                  AGE
daemonset.apps/apiserver  3         3         3       3            3           node-
role.kubernetes.io/master=  18h
```

5.11. 授予 DEDICATED-ADMIN 访问权限

只有创建集群的用户才能向其他 **cluster-admin** 或 **dedicated-admin** 用户授予集群访问权限。具有 **dedicated-admin** 访问权限的用户具有较少的特权。作为最佳实践，向大多数管理员授予 **dedicated-admin** 访问权限。

先决条件

- 您已在集群中添加身份提供程序(IDP)。
- 您有要创建的用户的 IDP 用户名。
- 已登陆到集群。

流程

1. 输入以下命令将用户提升为 **dedicated-admin** ：

```
$ rosa grant user dedicated-admin --user=<idp_user_name> --cluster=<cluster_name>
```

2. 输入以下命令验证您的用户现在是否有 **dedicated-admin** 访问权限：

```
$ oc get groups dedicated-admins
```

输出示例

```
NAME              USERS
dedicated-admins  rh-rosa-test-user
```

注意

如果没有 **dedicated-admin** 特权的用户运行此命令，则会显示 **Forbidden** 错误。

5.12. 未经身份验证的组的集群角色绑定

注意

在 Red Hat OpenShift Service on AWS 4.16 之前，未经身份验证的组可以访问一些集群角色。在 AWS 4.16 上的 Red Hat OpenShift Service on AWS 4.16 之前，从版本更新的集群会保留这个未经身份验证的组的访问权限。

为了安全起见，Red Hat OpenShift Service on AWS 4 不允许未经身份验证的组具有集群角色的默认访问权限。

在有些用例中，可能需要将 **system:unauthenticated** 添加到集群角色中。

集群管理员可以将未经身份验证的用户添加到以下集群角色中：

- **system:scope-impersonation**
- **system:webhook**
- **system:oauth-token-deleter**
- **self-access-reviewer**

重要

在修改未经身份验证的访问时，始终验证符合您机构的安全标准。

第 6 章 了解并创建服务帐户

6.1. 服务帐户概述

服务帐户是一个 Red Hat OpenShift Service on AWS 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。

当您使用 Red Hat OpenShift Service on AWS CLI 或 Web 控制台时，您的 API 令牌会向 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。

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。

6.2. 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-dockercfg-qzbhb
Tokens:              robot-token-f4khf
Events:              <none>
```

6.3. 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
```

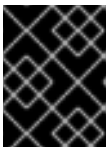
提示

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 7 章 在应用程序中使用服务帐户

7.1. 服务帐户概述

服务帐户是一个 Red Hat OpenShift Service on AWS 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。

当您使用 Red Hat OpenShift Service on AWS CLI 或 Web 控制台时，您的 API 令牌会向 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。

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。

7.2. 默认服务帐户

您的 Red Hat OpenShift Service on AWS 集群包含用于集群管理的默认服务帐户，并为每个项目生成更多服务帐户。

7.2.1. 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。在服务器启动时的 Red Hat OpenShift Service on AWS 基础架构项目(**openshift-infra**)中创建以下服务帐户，并授予集群范围的以下角色：

服务帐户	描述
replication-controller	分配 **system:replication-controller** 角色
deployment-controller	分配 **system:deployment-controller** 角色。

服务帐户	描述
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。

7.2.2. 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。 **注意** 如果没有启用 **Build** 集群功能，则不会创建 **builder** 服务帐户。
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。 **注意** 如果没有启用 **DeploymentConfig** 集群功能，则不会创建 **deployer** 服务帐户。
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。*

7.2.3. 自动生成的镜像 pull secret

默认情况下，Red Hat OpenShift Service on AWS 为每个服务帐户创建一个镜像 pull secret。

注意

在 Red Hat OpenShift Service on AWS 4.16 之前，还会为每个创建的服务帐户 API 令牌 secret 生成长期服务帐户令牌 secret。从 Red Hat OpenShift Service on AWS 4.16 开始，不再创建此服务帐户 API 令牌 secret。

升级到 4 后，任何现有的长期服务帐户 API 令牌 secret 不会被删除，并将继续正常工作。有关检测集群中使用的长期 API 令牌，以及在不需要时删除它们的信息，请参阅红帽知识库文章 OpenShift Container Platform 中的 Long-lived 服务帐户 API 令牌 。

此镜像 pull secret 需要将 OpenShift 镜像 registry 集成到集群的用户身份验证和授权系统中。

*但是，如果您不启用 **ImageRegistry** 功能，或者在 Cluster Image Registry Operator 配置中禁用集成的 OpenShift 镜像 registry，则不会为每个服务帐户生成镜像 pull secret。*

当在之前启用的集群中禁用集成的 OpenShift 镜像 registry 时，之前生成的镜像 pull secret 会被自动删除。

7.3. 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-dockercfg-qzbhb
Tokens:              robot-token-f4khf
Events:             <none>
```

第 8 章 使用服务帐户作为 OAUTH 客户端

8.1. 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。

8.1.1. 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 9 章 假设服务帐户的 AWS IAM 角色

在使用 AWS 安全令牌服务(STS)的 Red Hat OpenShift Service on AWS 集群中，OpenShift API 服务器可以启用为项目签名的服务帐户令牌，用于假定 pod 中的 AWS Identity and Access Management (IAM)角色。如果假定的 IAM 角色具有所需的 AWS 权限，pod 可以使用临时 STS 凭证对 AWS API 进行身份验证，以执行 AWS 操作。

您可以使用 pod 身份 Webhook 来项目服务帐户令牌，以假定您自己的工作负载的 AWS Identity and Access Management (IAM)角色。如果假定的 IAM 角色具有所需的 AWS 权限，pod 可以使用临时 STS 凭证运行 AWS SDK 操作。

9.1. 服务帐户如何假定 SRE 拥有的项目中的 AWS IAM 角色

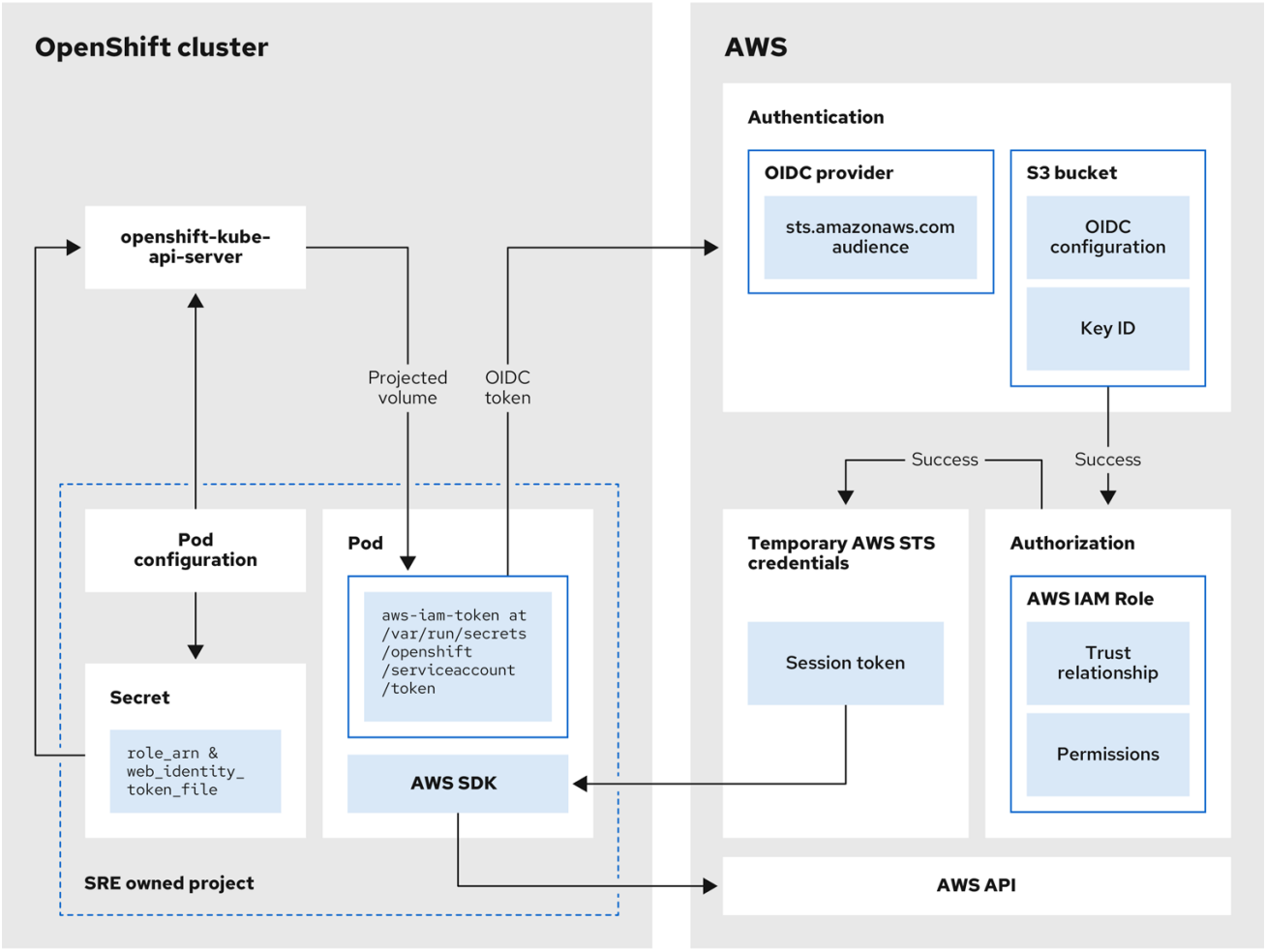
当使用 AWS 安全令牌服务(STS)的 AWS 集群上安装 Red Hat OpenShift Service 时，会创建特定于集群的 Operator AWS Identity and Access Management (IAM)角色。这些 IAM 角色允许 Red Hat OpenShift Service on AWS 集群 Operator 运行核心 OpenShift 功能。

集群 Operator 使用服务帐户假设 IAM 角色。当服务帐户假设 IAM 角色时，会为集群 Operator 的 pod 中使用的服务帐户提供临时 STS 凭证。如果假定角色具有所需的 AWS 权限，则服务帐户可以在 pod 中运行 AWS SDK 操作。

在 SRE 拥有的项目中假设 AWS IAM 角色的工作流

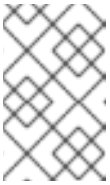
下图演示了在 SRE 拥有的项目中假设 AWS IAM 角色的工作流：

图 9.1. 在 SRE 拥有的项目中假设 AWS IAM 角色的工作流



工作流有以下阶段：

1. 在集群 Operator 运行的每个项目中，Operator 的部署 spec 具有投射服务帐户令牌的卷挂载，以及包含 pod 的 AWS 凭证配置的 secret。令牌是面向使用者和限时的。每小时，Red Hat OpenShift Service on AWS 会生成新的令牌，AWS SDK 会读取包含 AWS 凭证配置的挂载的 secret。此配置具有到挂载令牌的路径和 AWS IAM 角色 ARN。secret 的凭证配置包括：
    - 一个 **$AWS_ARN_ROLE** 变量，其中包含具有运行 AWS SDK 操作所需的权限的 IAM 角色的 ARN。
    - **$AWS_WEB_IDENTITY_TOKEN_FILE** 变量，在 pod 中具有到服务帐户的 OpenID Connect (OIDC) 令牌的完整路径。完整路径为 **/var/run/secrets/openshift/serviceaccount/token**。
2. 当集群 Operator 需要假设 AWS IAM 角色访问 AWS 服务（如 EC2）时，Operator 上运行的 AWS SDK 客户端代码会调用 **AssumeRoleWithWebIdentity** API 调用。
3. OIDC 令牌从 pod 传递给 OIDC 供应商。如果满足以下要求，供应商会验证服务帐户身份：
    - 身份签名由私钥有效并签名。
    - **sts.amazonaws.com** 受众列在 OIDC 令牌中，并与 OIDC 供应商中配置的受众匹配。

    注意

    在带有 STS 的 AWS 上的 Red Hat OpenShift Service 中，OIDC 供应商会在安装过程中创建，并默认设置为服务帐户签发者。在 OIDC 供应商中默认设置 **sts.amazonaws.com** 受众。

    - OIDC 令牌尚未过期。
    - 令牌中的签发者值具有 OIDC 供应商的 URL。
4. 如果项目和服务帐户位于被假定的 IAM 角色的信任策略范围内，则授权会成功。
5. 成功身份验证和授权后，临时 AWS STS 凭证以 AWS 访问令牌、secret 密钥和会话令牌的形式传递给 pod，供服务帐户使用。通过使用凭证，服务帐户被临时授予 IAM 角色中启用的 AWS 权限。
6. 当集群 Operator 运行时，使用 pod 中的 AWS SDK 的 Operator 会消耗具有投射服务帐户和 AWS IAM 角色 ARN 的 secret，以针对 OIDC 供应商进行身份验证。OIDC 供应商返回临时 STS 凭证，用于针对 AWS API 进行身份验证。

9.2. 服务帐户如何假定用户定义的项目中的 AWS IAM 角色

当使用 AWS 安全令牌服务 (STS) 在 AWS 集群上安装 Red Hat OpenShift Service 时，默认包括 Pod 身份 Webhook 资源。

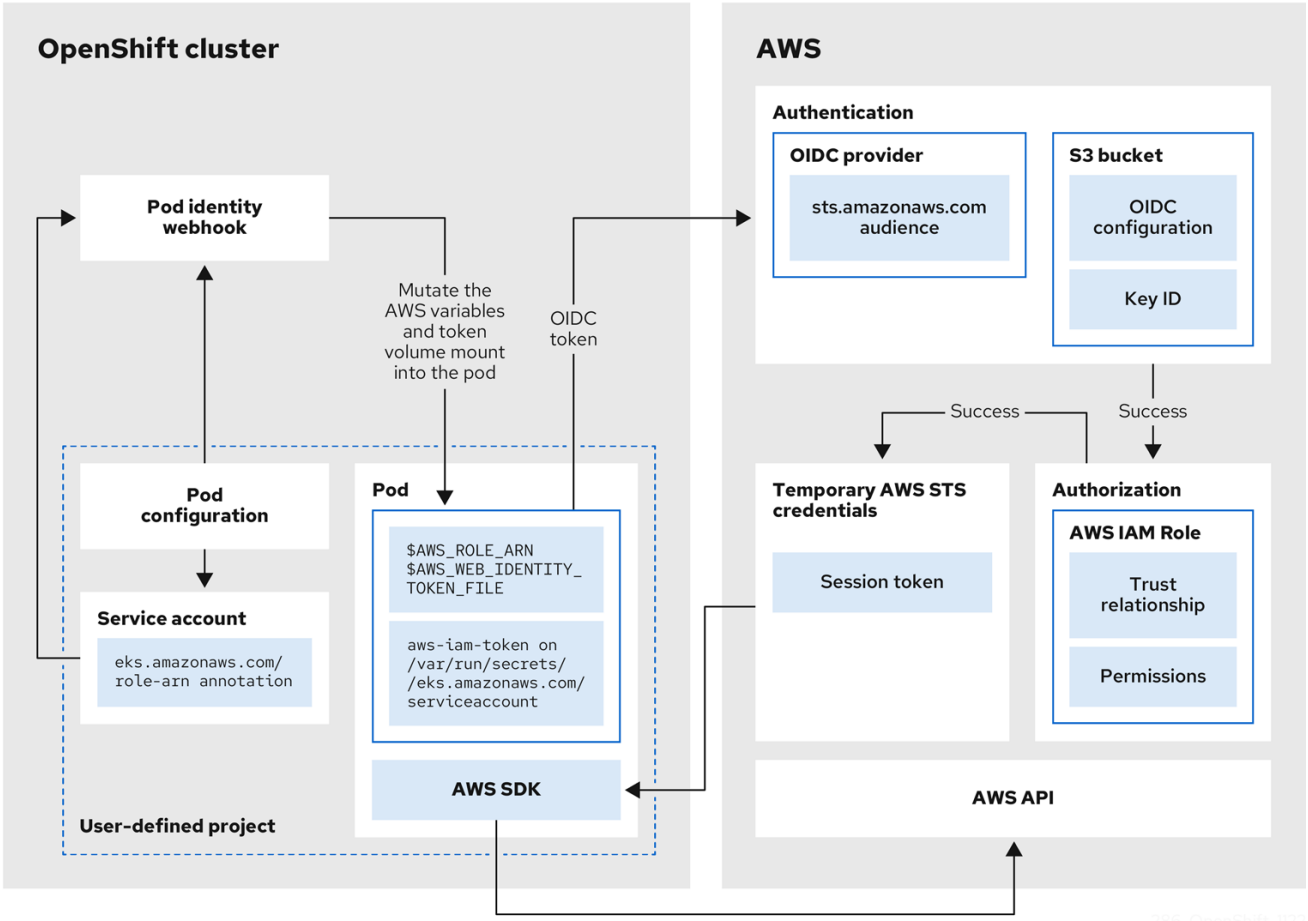
您可以使用 pod 身份 Webhook 在用户定义的项目中启用服务帐户，来假定同一项目中的 pod 中的 AWS Identity and Access Management (IAM) 角色。当假定 IAM 角色时，会为 pod 中的服务帐户提供临时 STS 凭证。如果假定角色具有所需的 AWS 权限，则服务帐户可以在 pod 中运行 AWS SDK 操作。

要为 pod 启用 pod 身份 Webhook，您必须在项目中使用 **eks.amazonaws.com/role-arn** 注解创建一个服务帐户。该注解必须引用您要假定服务帐户的 AWS IAM 角色的 Amazon Resource Name (ARN)。您还必须在 **Pod** 规格中引用服务帐户，并将 pod 部署到与服务帐户相同的项目中。

用户定义的项目中的 Pod 身份 Webhook 工作流

下图演示了用户定义的项目中的 pod 身份 Webhook 工作流：

图 9.2. 用户定义的项目中的 Pod 身份 Webhook 工作流



工作流有以下阶段：

1. 在用户定义的项目中，用户会创建一个包括 **eks.amazonaws.com/role-arn** 注解的服务帐户。该注解指向您希望服务帐户假定的 AWS IAM 角色的 ARN。

2. 当使用引用被注解服务帐户的配置在同一项目中部署 pod 时，pod 身份 Webhook 会修改 pod。变异会将以下组件注入 pod，而无需在 **Pod** 或 **Deployment** 资源配置中指定它们：

   - **$AWS_ARN_ROLE** 环境变量，其中包含 IAM 角色的 ARN，该角色具有运行 AWS SDK 操作所需的权限。

   - **$AWS_WEB_IDENTITY_TOKEN_FILE** 环境变量，其中包含服务帐户的 OpenID Connect (OIDC) 令牌的完整路径。完整路径为 **/var/run/secrets/eks.amazonaws.com/serviceaccount/token**。

   - 挂载到挂载点 **/var/run/secrets/eks.amazonaws.com/serviceaccount** 上的 **aws-iam-token** 卷。名为 **token** 的 OIDC 令牌文件包含在卷中。

3. OIDC 令牌从 pod 传递给 OIDC 供应商。如果满足以下要求，供应商会验证服务帐户身份：

   - 身份签名由私钥有效并签名。

   - **sts.amazonaws.com** 受众列在 OIDC 令牌中，并与 OIDC 供应商中配置的受众匹配。

注意

默认情况下，pod 身份 Webhook 会将 **sts.amazonaws.com** 受众应用到 OIDC 令牌。

在带有 STS 的 AWS 上的 Red Hat OpenShift Service 中，OIDC 供应商会在安装过程中创建，并默认设置为服务帐户签发者。在 OIDC 供应商中默认设置 **sts.amazonaws.com** 受众。

- OIDC 令牌尚未过期。
- 令牌中的签发者值包含 OIDC 供应商的 URL。

4. 如果项目和服务帐户位于被假定的 IAM 角色的信任策略范围内，则授权会成功。
5. 成功身份验证和授权后，以会话令牌的形式将临时 AWS STS 凭证传递给 pod，供服务帐户使用。通过使用凭证，服务帐户被临时授予 IAM 角色中启用的 AWS 权限。
6. 当您在 pod 中运行 AWS SDK 操作时，服务帐户为 AWS API 提供临时 STS 凭证来验证其身份。

9.3. 假设您自己的 POD 中的 AWS IAM 角色

按照本节中的步骤，启用服务帐户在用户定义的项目中部署的 pod 中假定 AWS Identity and Access Management (IAM) 角色。

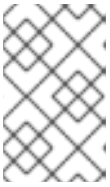
您可以创建所需资源，包括 AWS IAM 角色、服务帐户、包括 AWS SDK 的容器镜像，以及使用镜像部署的 pod。在示例中，使用了 Python 的 AWS Boto3 SDK。您还可以验证 pod 身份 Webhook 是否将 AWS 环境变量、卷挂载和令牌卷放入 pod。另外，您可以检查服务帐户是否在 pod 中假设 AWS IAM 角色，并可成功运行 AWS SDK 操作。

9.3.1. 为服务帐户设置 AWS IAM 角色

创建一个 AWS Identity and Access Management (IAM) 角色，由 Red Hat OpenShift Service on AWS 集群中的服务帐户假定。附加您的服务帐户在 pod 中运行 AWS SDK 操作所需的权限。

先决条件

- 您有在 AWS 帐户中安装和配置 IAM 角色所需的权限。
- 您可以访问使用 AWS 安全令牌服务 (STS) 的 Red Hat OpenShift Service on AWS 集群。不需要 admin 级别用户权限。
- 您有 OpenID Connect (OIDC) 供应商的 Amazon Resource Name (ARN)，该供应商被配置为 Red Hat OpenShift Service on AWS 中的服务帐户签发者。



注意

在带有 STS 的 AWS 上的 Red Hat OpenShift Service 中，OIDC 供应商会在安装过程中创建，并默认设置为服务帐户签发者。如果您不知道 OIDC 供应商 ARN，请联络您的集群管理员。

- 已安装 AWS CLI (**aws**)。

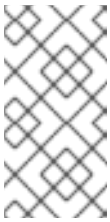
流程

1. 使用以下 JSON 配置创建一个名为 **trust-policy.json** 的文件：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "<oidc_provider_arn>" 1
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "<oidc_provider_name>:sub": "system:serviceaccount:<project_name>:
<service_account_name>" 2
                }
            }
        }
    ]
}
```

1 将 **<oidc_provider_arn>** 替换为 OIDC 供应商的 ARN，如 **arn:aws:iam::<aws_account_id>:oidc-provider/rh-oidc.s3.us-east-1.amazonaws.com/1v3r0n44npxu4g58so46aeohduomfres**。

2 将角色限制为指定的项目和服务帐户。将 **<oidc_provider_name>** 替换为 OIDC 供应商的名称，如 **rh-oidc.s3.us-east-1.amazonaws.com/1v3r0n44npxu4g58so46aeohduomfres**。将 **<project_name>:<service_account_name>** 替换为您的项目名称和服务帐户名称，如 **my-project:test-service-account**。



注意

另外，您可以使用 "**< oidc_provider_name>:sub": "system:serviceaccount:<project_name>:***"，将角色限制为指定项目内的任何服务帐户。如果提供 * 通配符，则必须在上一行中将 **StringEquals** 替换为 **StringLike**。

2. 创建一个 AWS IAM 角色，它使用 **trust-policy.json** 文件中定义的信任策略：

```
$ aws iam create-role \
  --role-name <aws_iam_role_name> \ 1
  --assume-role-policy-document file://trust-policy.json 2
```

1 将 **<aws_iam_role_name>** 替换为 IAM 角色的名称，如 **pod-identity-test-role**。

2 引用您在上一步中创建的 **trust-policy.json** 文件。

输出示例：

```
ROLE    arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name>      2022-09-
28T12:03:17+00:00      /      AQWMS3TB4Z2N3SH7675JK   <aws_iam_role_name>
```

```
ASSUMEROLEPOLICYDOCUMENT        2012-10-17
STATEMENT       sts:AssumeRoleWithWebIdentity   Allow
STRINGEQUALS    system:serviceaccount:<project_name>:<service_account_name>
PRINCIPAL       <oidc_provider_arn>
```

在输出中为角色保留 ARN。角色 ARN 的格式为 **arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name>**。

3. 附加服务帐户在 pod 中运行 AWS SDK 操作时所需的受管 AWS 权限：

```
$ aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \ 1
    --role-name <aws_iam_role_name> 2
```

1 本例中的策略为 IAM 角色添加了只读权限。

2 将 < **aws_iam_role_name** > 替换为您在上一步中创建的 IAM 角色的名称。

4. 可选：在角色中添加自定义属性或权限边界。如需更多信息，请参阅 AWS 文档中的创建角色将权限委派给 AWS 服务。

9.3.2. 在项目中创建服务帐户

在用户定义的项目中添加服务帐户。在服务帐户配置中包含 **eks.amazonaws.com/role-arn** 注解，该注解引用 AWS Identity and Access Management (IAM) 角色的 Amazon Resource Name (ARN) 角色。

先决条件

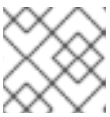
- 您已为您的服务帐户创建 AWS IAM 角色。如需更多信息，请参阅为服务帐户设置 AWS IAM 角色。
- 您可以使用 AWS 安全令牌服务 (STS) 集群访问 Red Hat OpenShift Service on AWS。不需要 admin 级别用户权限。
- 已安装 OpenShift CLI(**oc**)。

流程

1. 在 Red Hat OpenShift Service on AWS 集群中，创建一个项目：

```
$ oc new-project <project_name> 1
```

1 将 **<project_name>** 替换为项目的名称。名称必须与您在 AWS IAM 角色配置中指定的项目名称匹配。



注意

在创建时，您将自动切换到项目。

2. 使用以下服务帐户配置，创建名为 **test-service-account.yaml** 的文件：

```
apiVersion: v1
```

```
kind: ServiceAccount
metadata:
  name: <service_account_name> 1
  namespace: <project_name> 2
  annotations:
    eks.amazonaws.com/role-arn: "<aws_iam_role_arn>" 3
```

1 将 **<service_account_name>** 替换为服务帐户的名称。名称必须与您在 AWS IAM 角色配置中指定的服务帐户名称匹配。

2 将 **<project_name>** 替换为项目的名称。名称必须与您在 AWS IAM 角色配置中指定的项目名称匹配。

3 指定服务帐户在 pod 中使用的 AWS IAM 角色的 ARN。将 **<aws_iam_role_arn>** 替换为您为服务帐户创建的 AWS IAM 角色的 ARN。角色 ARN 的格式为 **arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name>**。

3. 在项目中创建服务帐户：

```
$ oc create -f test-service-account.yaml
```

输出示例：

```
serviceaccount/<service_account_name> created
```

4. 查看服务帐户的详情：

```
$ oc describe serviceaccount <service_account_name> 1
```

1 将 **<service_account_name>** 替换为服务帐户的名称。

输出示例：

```
Name:                <service_account_name> 1
Namespace:           <project_name> 2
Labels:              <none>
Annotations:         eks.amazonaws.com/role-arn: <aws_iam_role_arn> 3
Image pull secrets:  <service_account_name>-dockercfg-rnjkq
Mountable secrets:   <service_account_name>-dockercfg-rnjkq
Tokens:              <service_account_name>-token-4gbjp
Events:              <none>
```

1 指定服务帐户的名称。

2 指定包含服务帐户的项目。

3 列出服务帐户假定的 AWS IAM 角色的 ARN 注解。

9.3.3. 创建 AWS SDK 容器镜像示例

此流程中的步骤提供了创建包含 AWS SDK 的容器镜像的示例方法。

示例步骤使用 Podman 创建容器镜像和 Quay.io 来托管该镜像。有关 Quay.io 的更多信息，请参阅 Quay.io 入门。容器镜像可用于部署可运行 AWS SDK 操作的 pod。

注意

在本例中，Python 的 AWS Boto3 SDK 安装到容器镜像中。有关安装和使用 AWS Boto3 SDK 的更多信息，请参阅 AWS Boto3 文档。有关其他 AWS SDK 的详情，请参阅 AWS 文档中的 AWS SDK 和工具参考指南。

先决条件

- 您已在安装主机上安装了 Podman。
- 您有一个 Quay.io 用户帐户。

流程

1. 将以下配置添加到名为 **Containerfile** 的文件中：

```
FROM ubi9/ubi (1)
RUN dnf makecache && dnf install -y python3-pip && dnf clean all && pip3 install
boto3>=1.15.0 (2)
```

(1) 指定 Red Hat Universal Base Image 版本 9。

(2) 使用 **pip** 软件包管理系统安装 AWS Boto3 SDK。在本例中，安装了 AWS Boto3 SDK 版本 1.15.0 或更高版本。

2. 在包含文件的目录中，构建名为 **awsboto3sdk** 的容器镜像：

```
$ podman build -t awsboto3sdk .
```

3. 登录到 Quay.io：

```
$ podman login quay.io
```

4. 标记镜像以准备上传到 Quay.io：

```
$ podman tag localhost/awsboto3sdk quay.io/<quay_username>/awsboto3sdk:latest (1)
```

(1) 将 **<quay_username&** gt; 替换为您的 Quay.io 用户名。

5. 将标记的容器镜像推送到 Quay.io：

```
$ podman push quay.io/<quay_username>/awsboto3sdk:latest (1)
```

(1) 将 **<quay_username&** gt; 替换为您的 Quay.io 用户名。

6. 创建包含镜像 public 的 Quay.io 存储库。这会发布镜像，以便使用它来在 Red Hat OpenShift Service on AWS 集群中部署 pod：

   a. 在 https://quay.io/ 上，导航到包含该镜像的存储库的 **Repository Settings** 页面。

   b. 单击 **Make Public**，使存储库公开可用。

9.3.4. 部署包含 AWS SDK 的 pod

从包含 AWS SDK 的容器镜像，在用户定义的项目中部署 pod。在 pod 配置中，指定包含 **eks.amazonaws.com/role-arn** 注解的服务帐户。

随着服务帐户引用 pod，pod 身份 Webhook 会将 AWS 环境变量、卷挂载和令牌卷注入 pod。pod 变异可让服务帐户在 pod 中自动假设 AWS IAM 角色。

先决条件

- 您已为服务帐户创建了 AWS Identity and Access Management (IAM)角色。如需更多信息，请参阅为服务帐户设置 AWS IAM 角色。
- 您可以访问使用 AWS 安全令牌服务(STS)的 Red Hat OpenShift Service on AWS 集群。不需要 admin 级别用户权限。
- 已安装 OpenShift CLI(**oc**)。
- 您已在项目中创建了一个服务帐户，其中包含一个 **eks.amazonaws.com/role-arn** 注解，该注解引用您想要服务帐户的 IAM 角色的 Amazon Resource Name (ARN)。
- 您有一个包含 AWS SDK 的容器镜像，且镜像可供集群使用。具体步骤请参阅创建示例 AWS SDK 容器镜像。

注意

在这个示例中使用 Python 的 AWS Boto3 SDK。有关安装和使用 AWS Boto3 SDK 的更多信息，请参阅 AWS Boto3 文档。有关其他 AWS SDK 的详情，请参阅 AWS 文档中的 AWS SDK 和工具参考指南。

流程

1. 使用以下 pod 配置创建一个名为 **awsboto3sdk-pod.yaml** 的文件：

```
apiVersion: v1
kind: Pod
metadata:
  namespace: <project_name> 1
  name: awsboto3sdk 2
spec:
  securityContext:
    runAsNonRoot: true
    seccompProfile:
      type: RuntimeDefault
  serviceAccountName: <service_account_name> 3
  containers:
  - name: awsboto3sdk
    image: quay.io/<quay_username>/awsboto3sdk:latest 4
```

```
    command:
    - /bin/bash
    - "-c"
    - "sleep 100000" 5
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: [ALL]
  terminationGracePeriodSeconds: 0
  restartPolicy: Never
```

1. 将 **<project_name>** 替换为项目的名称。名称必须与您在 AWS IAM 角色配置中指定的项目名称匹配。
2. 指定 pod 的名称。
3. 将 **<service_account_name>** 替换为配置为假定 AWS IAM 角色的服务帐户的名称。名称必须与您在 AWS IAM 角色配置中指定的服务帐户名称匹配。
4. 指定 **awsboto3sdk** 容器镜像的位置。将 **<quay_username&** gt; 替换为您的 Quay.io 用户名。
5. 在本例中，此行使 pod 保持运行 100000 秒，以便在 pod 中直接启用验证测试。如需详细的验证步骤，请参阅 pod 中验证假定的 IAM 角色。

2. 部署 **awsboto3sdk** pod:

```
$ oc create -f awsboto3sdk-pod.yaml
```

输出示例：

```
pod/awsboto3sdk created
```

9.3.5. 在 pod 中验证假定的 IAM 角色

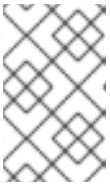
在项目中部署 **awsboto3sdk** pod 后，验证 pod 身份 Webhook 是否已修改 pod。检查 pod 中是否存在所需的 AWS 环境变量、卷挂载和 OIDC 令牌卷。

您还可以在 pod 中运行 AWS SDK 操作时，验证服务帐户是否假定 AWS 帐户的 AWS Identity and Access Management (IAM) 角色。

先决条件

- 您已为您的服务帐户创建 AWS IAM 角色。如需更多信息，请参阅为服务帐户设置 AWS IAM 角色。
- 您可以访问使用 AWS 安全令牌服务(STS)的 Red Hat OpenShift Service on AWS 集群。不需要 admin 级别用户权限。
- 已安装 OpenShift CLI(**oc**)。
- 您已在项目中创建了一个服务帐户，其中包含一个 **eks.amazonaws.com/role-arn** 注解，该注解引用您想要服务帐户的 IAM 角色的 Amazon Resource Name (ARN)。

- 您已在包含 AWS SDK 的用户定义的项目中部署了 pod。pod 引用了使用 pod 身份 Webhook 的服务帐户，以假设运行 AWS SDK 操作所需的 AWS IAM 角色。具体步骤请参阅部署包含 AWS SDK 的 pod。



注意

在本例中，使用包括 Python 的 AWS Boto3 SDK 的 pod。有关安装和使用 AWS Boto3 SDK 的更多信息，请参阅 AWS Boto3 文档。有关其他 AWS SDK 的详情，请参阅 AWS 文档中的 AWS SDK 和工具参考指南。

流程

1. 验证 AWS 环境变量、卷挂载和 OIDC 令牌卷是否在部署的 **awsboto3sdk** pod 的描述中列出：

```
$ oc describe pod awsboto3sdk
```

输出示例：

```
Name:         awsboto3sdk
Namespace:    <project_name>
...
Containers:
  awsboto3sdk:
    ...
    Environment:
      AWS_ROLE_ARN:                 <aws_iam_role_arn> 1
      AWS_WEB_IDENTITY_TOKEN_FILE:
/var/run/secrets/eks.amazonaws.com/serviceaccount/token 2
    Mounts:
      /var/run/secrets/eks.amazonaws.com/serviceaccount from aws-iam-token (ro) 3
...
Volumes:
  aws-iam-token: 4
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   86400
...
```

1 列出由 pod 身份 webhook 注入 pod 的 **AWS_ROLE_ARN** 环境变量。变量包含服务帐户假定的 AWS IAM 角色的 ARN。

2 列出由 pod 身份 Webhook 注入的 **AWS_WEB_IDENTITY_TOKEN_FILE** 环境变量。变量包含用于验证服务帐户身份的 OIDC 令牌的完整路径。

3 列出 Pod 身份 Webhook 注入 pod 的卷挂载。

4 列出挂载到 **/var/run/secrets/eks.amazonaws.com/serviceaccount** 挂载点的 **aws-iam-token** 卷。卷包含用于验证服务帐户的 OIDC 令牌，以假定 AWS IAM 角色。

2. 在 **awsboto3sdk** pod 中启动一个交互式终端：

```
$ oc exec -ti awsboto3sdk -- /bin/sh
```

3. 在 pod 的交互式终端中，验证 pod 身份 webhook 是否将 **$AWS_ROLE_ARN** 环境变量放入 pod 中：

```
$ echo $AWS_ROLE_ARN
```

输出示例：

```
arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name> 1
```

1 输出必须指定 AWS IAM 角色的 ARN，该角色具有运行 AWS SDK 操作所需的权限。

4. 在 pod 的交互式终端中，验证 **$AWS_WEB_IDENTITY_TOKEN_FILE** 环境变量是否被 pod 身份 webhook 放入 pod 中：

```
$ echo $AWS_WEB_IDENTITY_TOKEN_FILE
```

输出示例：

```
/var/run/secrets/eks.amazonaws.com/serviceaccount/token 1
```

1 输出必须为服务帐户指定 pod 中到 OIDC 令牌的完整路径。

5. 在 pod 的交互式终端中，验证包含 OIDC 令牌文件的 **aws-iam-token** 卷挂载是否由 pod 身份 webhook 挂载：

```
$ mount | grep -is 'eks.amazonaws.com'
```

输出示例：

```
tmpfs on /run/secrets/eks.amazonaws.com/serviceaccount type tmpfs
(ro,relatime,seclabel,size=13376888k)
```

6. 在 pod 的交互式终端中，验证 **/var/run/secrets/eks.amazonaws.com/serviceaccount/** 挂载点中是否存在名为 **token** 的 OIDC 令牌文件：

```
$ ls /var/run/secrets/eks.amazonaws.com/serviceaccount/token
```

输出示例：

```
/var/run/secrets/eks.amazonaws.com/serviceaccount/token 1
```

1 **aws-iam-token** 卷中的 OIDC 令牌文件，它由 pod 身份 Webhook 挂载到 pod 中。令牌用于验证 AWS 中服务帐户的身份。

7. 在 pod 中，验证 AWS Boto3 SDK 操作是否已成功运行：

   a. 在 pod 的交互式终端中，启动一个 Python 3 shell：

```
$ python3
```

b. 在 Python 3 shell 中导入 **boto3** 模块：

```
>>> import boto3
```

c. 创建包含 Boto3 **s3** 服务资源的变量：

```
>>> s3 = boto3.resource('s3')
```

d. 打印 AWS 帐户中的所有 S3 存储桶的名称：

```
>>> for bucket in s3.buckets.all():
...     print(bucket.name)
...
```

输出示例：

```
<bucket_name>
<bucket_name>
<bucket_name>
...
```

如果服务帐户成功假定 AWS IAM 角色，则输出会列出 AWS 帐户中提供的所有 S3 存储桶。

9.4. 其他资源

- 有关在服务帐户中使用 AWS IAM 角色的更多信息，请参阅 AWS 文档中的服务帐户的 IAM 角色。
- 有关 AWS IAM 角色委托的详情，请参阅 AWS 文档中的创建角色将权限委派给 AWS 服务。
- 有关 AWS SDK 的详情，请参阅 AWS 文档中的 AWS SDK 和工具参考指南。
- 有关为 Python 安装和使用 AWS Boto3 SDK 的更多信息，请参阅 AWS Boto3 文档。
- 如需有关 OpenShift 的 webhook 准入插件的常规信息，请参阅 OpenShift Container Platform 文档中的 Webhook 准入插件。

第 10 章 界定令牌作用域

10.1. 关于界定令牌作用域

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。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 **dedicated-admin** 角色的用户才能创建有作用域令牌。

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。

10.1.1. 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。

10.1.2. 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。

10.2. 将未经身份验证的组添加到集群角色中

作为集群管理员，您可以通过创建集群角色绑定，将未经身份验证的用户添加到 Red Hat OpenShift Service on AWS 中的以下集群角色中。未经身份验证的用户无法访问非公共集群角色。这只应在需要时在特定用例中完成。

您可以将未经身份验证的用户添加到以下集群角色中：

- **system:scope-impersonation**
- **system:webhook**

- ***system:oauth-token-deleter***
- ***self-access-reviewer***

重要

在修改未经身份验证的访问时，始终验证符合您机构的安全标准。

先决条件

- *您可以使用具有 **cluster-admin** 角色的用户访问集群。*
- *已安装 OpenShift CLI(**oc**)。*

流程

1. *创建名为 **add-<cluster_role>-unauth.yaml** 的 YAML 文件，并添加以下内容：*

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 annotations:
   rbac.authorization.kubernetes.io/autoupdate: "true"
 name: <cluster_role>access-unauthenticated
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: <cluster_role>
subjects:
 - apiGroup: rbac.authorization.k8s.io
   kind: Group
   name: system:unauthenticated
```

2. *运行以下命令来应用配置：*

```
$ oc apply -f add-<cluster_role>.yaml
```

第 11 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，这可以提高与云供应商身份访问管理(IAM)服务集成的功能，如 AWS IAM 或 Google Cloud Platform IAM 上的 Red Hat OpenShift Service。

11.1. 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。

11.2. 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 **dedicated-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. 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:
  securityContext:
    runAsNonRoot: true (1)
    seccompProfile:
      type: RuntimeDefault (2)
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /var/run/secrets/tokens
      name: vault-token
    securityContext:
      allowPrivilegeEscalation: false
      capabilities:
        drop: [ALL]
  serviceAccountName: build-robot (3)
  volumes:
  - name: vault-token
    projected:
      sources:
      - serviceAccountToken:
```

```
      path: vault-token 4
      expirationSeconds: 7200 5
      audience: vault 6
```

1 防止容器以 root 用户身份运行，以最大程度降低威胁的风险。

2 设置默认 seccomp 配置集，限制为仅可以使用必要的系统调用，以降低风险。

3 对现有服务帐户的引用。

4 相对于文件挂载点的相对路径，用于将令牌放入。

5 （可选）设置服务帐户令牌的到期时间（以秒为单位）。默认值为 3600 秒（1 小时），这个值必须至少为 600 秒（10 分钟）。如果令牌已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。

6 （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。

注意

为了防止意外失败，Red Hat OpenShift Service on AWS 会使用 **--service-account-extend-token-expiration** 默认值从初始令牌生成中覆盖了 **expirationSeconds** 值。**您无法更改此设置。**

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。

2. 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。

11.3. 在 POD 外部创建绑定服务帐户令牌

先决条件

- 您可以使用具有 dedicated-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

- 运行以下命令，在 pod 外部创建绑定服务帐户令牌：

```
$ oc create token build-robot
```

输出示例

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IkY2M1N4MHRvc2xFNnFSQlA4eG9GYzVPdnN3NkhIV0tRWmFrUDRNcWx4S0kifQ.eyJhdWQiOlsiaHR0cHM6Ly9pc3N1ZXIyLnRlc3QuY29tIiwiaHR0cHM6Ly9pc3N1ZXIxLnRlc3QuY29tIiwiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjIl0sImV4cCI6MTY3OTU0MzgzMCwiaWF0IjoxNjc5NTQwMjMwLCJpc3MiOiJodHRwczovL2lzc3VlcjludGVzdC5jb20iLCJrdWJlcm5ldGVzLmlvIjp7Im5hbWVzcGFjZSI6ImRlZmF1bHQiLCJzZXJ2aWNlYWNjb3VudCI6eyJuYW1lIjoidGVzdC1zYSIsInVpZCI6ImM3ZjA4MjkwLWIzOTUtNGM4NC04NjI4LTMzMTM1NTVhNWY1OSJ9fSwibmJmIjoxNjc5NTQwMjMwLCJzdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6ZGVmYXVsdDp0ZXN0LXNhIn0.WyAOPvh1BFMUl3LNhBCrQeaB5wSynbnCfojWuNNPSilT4YvFnKibxwREwmzHpV4LO1xOFZHSi6bXBOmG_o-m0XNDYL3FrGHd65mymiFyluztxa2lgHVxjw5reIV5ZLgNSol3Y8bJqQqmNg3rtQQWRML2kpJBXdDHNww0E5XOypmffYkfkadli8lN5QQD-MhsCbiAF8waCYs8bj6V6Y7uUKTcxee8sCjiRMVtXKjQtooERKm-CH_p57wxCljIBeM89VdaR51NJGued4hVV5lxvVrYZFu89lBEAq4oyQN_d6N1vBWGXQMyoihnt_fQjn-NfnlJWk-3NSZDlluDJAv7e-MTEk3geDrHVQKNEzDei2-Un64hSzb-n1g1M0Vn0885wQBQAePC9UlZm8YZlMNk1tq6wlUKQTMv3HPfi5HtBRqVc2eVs0EfMX4-x-PHhPCasJ6qLJWyj6DvyQ08dP4DW_TWZVGvKlmld0hzwpg59TTcLR0iCklSEJgAVEEd13Aa_M0-faD11L3MhUGxw0qxgOsPczdXUsolSlSbefs7OKymzFSlkTAn9sDQ8PHMOsuyxsK8vzfrR-E0z7MAeguZ2kaIY7cZqbN6WFy0caWgx46hrKem9vCKALefElRYbCg3hcBmowBcRTOqaFHLNnHghhU1LaRpoFzH7OUarqX9SGQ
```

其他资源

- 创建服务帐户

第 12 章 管理安全性上下文约束

在 Red Hat OpenShift Service on AWS 中，您可以使用安全性上下文约束(SCC)来控制集群中 pod 的权限。

安装期间会创建默认 SCC，安装一些 Operator 或其他组件。作为集群管理员，您还可以使用 OpenShift CLI (oc)创建自己的 SCC。

重要

不要修改默认 SCC。自定义默认 SCC 可能会导致在某些平台 Pod 部署或 ROSA 升级时出现问题。另外，默认 SCC 值会在一些集群升级过程中重置为默认值，这会丢弃对这些 SCC 的所有自定义。

根据需要创建并修改您自己的 SCC，而不是修改默认 SCC。有关详细步骤，请参阅创建安全性上下文约束。

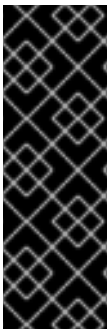
12.1. 关于安全性上下文约束

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。这些权限决定了 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 Red Hat OpenShift Service on AWS 中的任何命名空间上设置 openshift.io/run-level 标签。此标签供内部 Red Hat OpenShift Service on AWS 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。

12.1.1. 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 Red Hat OpenShift Service on AWS 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 可能会导致在某些平台 Pod 部署或 ROSA 升级时出现问题。另外，默认 SCC 值会在一些集群升级过程中重置为默认值，这会丢弃对这些 SCC 的所有自定义。

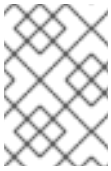
根据需要创建并修改您自己的 SCC，而不是修改默认 SCC。有关详细步骤，请参阅 创建安全性上下文约束。

表 12.1. 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。

安全性上下文约束	描述
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，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。
hostnetwork-v2	与 **hostnetwork** SCC 类似，但有以下区别： • **ALL** 能力会从容器中丢弃。 • 可以显式添加 **NET_BIND_SERVICE** 功能。 • 默认情况下，**seccompProfile** 设置为 **runtime/default**。 • 在安全上下文中，需要取消设置 **allowPrivilegeEscalation** 或将其设为 **false**。
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。

安全性上下文约束	描述
nonroot-v2	与 **nonroot** SCC 类似，但有以下不同： • **ALL** 能力会从容器中丢弃。 • 可以显式添加 **NET_BIND_SERVICE** 功能。 • 默认情况下，**seccompProfile** 设置为 **runtime/default**。 • 在安全上下文中，需要取消设置 **allowPrivilegeEscalation** 或将其设为 **false**。
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。 **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 ● 要求 pod 使用预分配的 FSGroup 运行 ● 允许 pod 使用任何补充组 在从 Red Hat OpenShift Service on AWS 4.10 或更早版本升级的集群中，任何经过验证的用户都可以使用这个 SCC。新 Red Hat OpenShift Service on AWS 4.11 或更高版本的安装用户不再提供 **restricted** SCC，除非明确授予了访问权限。
restricted-v2	与 **root** SCC 类似，但有以下不同： ● **ALL** 能力会从容器中丢弃。 ● 可以显式添加 **NET_BIND_SERVICE** 功能。 ● 默认情况下，**seccompProfile** 设置为 **runtime/default**。 ● 在安全上下文中，需要取消设置 **allowPrivilegeEscalation** 或将其设为 **false**。 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。  **注意** **restricted-v2** SCC 是系统默认包括的最严格的 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，将 **readOnlyRootFilesystem** 限制为 **true**。

12.1.2. 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**。

类别	描述
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。这是 restricted-v2 SCC 的作用。

12.1.3. 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。*

12.1.4. 控制卷

通过设置 SCC 的 volumes 字段，控制特定卷类型的使用。

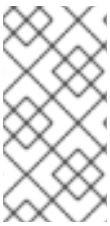
此字段的允许值与创建卷时定义的卷来源对应：

- *awsElasticBlockStore*
- *azureDisk*
- *azureFile*
- *cephFS*
- *cinder*
- *configMap*
- *csi*
- *downwardAPI*

- *emptyDir*
- *fc*
- *flexVolume*
- *flocker*
- *gcePersistentDisk*
- *ephemeral*
- *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 Red Hat OpenShift Service on AWS 时都会添加新的类型。

注意

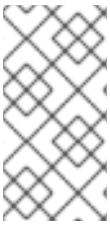
为向后兼容，使用 allowHostDirVolumePlugin 将覆盖 volumes 字段中的设置。例如，如果 allowHostDirVolumePlugin 设为 false，但在 volumes 字段中是允许，则将移除 volumes 中的 hostPath 值。

12.1.5. 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。



注意

在创建工作负载资源（如部署）时，只有服务帐户用于查找 SCC，并在创建时接受 Pod。

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。

12.1.6. 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强的限制进行排序。*

3. 如果优先级和限制都相等，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。

12.2. 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：

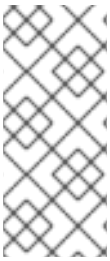
1. RunAsAny 和 MustRunAsNonRoot 策略不提供默认值。如果 Pod 需要参数值，如组 ID，您必须在 Pod 规格中定义这个值。

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。

12.3. 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: 10
- '*'
```

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。

10

安全上下文允许的卷类型。在这个示例中，* 允许使用所有卷类型。

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都授予了访问 restricted-v2 的权限。

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

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，Red Hat OpenShift Service on AWS 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。

12.4. 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）。

重要

创建和修改您自己的 SCC 是可能导致集群不稳定的高级操作。如果您对使用自己的 SCC 有疑问，请联系红帽支持。有关联系红帽支持的详情，请参考 获取支持。

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。

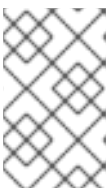
流程

1. 在名为 scc-admin.yaml 的 YAML 文件中定义 SCC：

```
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]
```

12.5. 配置工作负载以要求特定的 SCC

您可以将工作负载配置为需要特定的安全性上下文约束 (SCC)。当您要将特定 SCC 固定到工作负载，或者要防止集群中被另一个 SCC 抢占时，这非常有用。

如果要求特定的 SCC，在工作负载上设置 openshift.io/required-scc 注解。您可以在可设置 pod 清单模板的任何资源上设置此注解，如部署或守护进程集。

SCC 必须存在于集群中，且必须适用于工作负载，否则 pod 准入会失败。如果用户创建了 Pod，或 Pod 的服务帐户的具有 pod 命名空间中的 SCC 的 use 权限，则 SCC 被视为适用于工作负载。

警告

不要更改 live pod 清单中的 openshift.io/required-scc 注解，因为这样做会导致 pod 准入失败。要更改所需的 SCC，请更新底层 Pod 模板中的注解，这会导致 pod 被删除并重新创建。

先决条件

- *集群中必须存在 SCC。*

流程

1. *为部署创建 YAML 文件，并通过设置 openshift.io/required-scc 注解来指定所需的 SCC：*

deployment.yaml 示例

```
apiVersion: config.openshift.io/v1
kind: Deployment
apiVersion: apps/v1
spec:
# ...
  template:
    metadata:
      annotations:
        openshift.io/required-scc: "my-scc" 1
# ...
```

1. *指定需要的 SCC 的名称。*

2. *运行以下命令来创建资源：*

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

验证

- 验证部署使用指定的 SCC：

a. 运行以下命令，查看 pod 的 openshift.io/scc 注解的值：

```
$ oc get pod <pod_name> -o jsonpath='{.metadata.annotations.openshift\.io\/scc}
{"\n"}' 1
```

1 将 <pod_name> 替换为部署 pod 的名称。

b. 检查输出，并确认显示的 SCC 与您在部署中定义的 SCC 匹配：

输出示例

```
my-scc
```

12.6. 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。

重要

不要在默认项目中运行工作负载或共享对默认项目的访问权限。为运行核心集群组件保留默认项目。

以下默认项目被视为具有高度特权：default, kube-public, kube-system, openshift, openshift-infra, openshift-node，其他系统创建的项目的标签 openshift.io/run-level 被设置为 0 或 1。依赖于准入插件（如 pod 安全准入、安全性上下文约束、集群资源配额和镜像引用解析）的功能无法在高特权项目中工作。

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-v2 SCC。

12.7. 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。

12.7.1. 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   <no value>              MustRunAs   RunAsAny           RunAsAny    RunAsAny    10           false            ["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
hostaccess              false   <no value>              MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <no value>   false            ["configMap","downwardAPI","emptyDir","hostPath","persistentVolumeClaim","projected","secret"]
hostmount-anyuid        false   <no value>              MustRunAs   RunAsAny           RunAsAny    RunAsAny    <no value>   false            ["configMap","downwardAPI","emptyDir","hostPath","nfs","persistentVolumeClaim","projected","secret"]
hostnetwork             false   <no value>              MustRunAs   MustRunAsRange     MustRunAs   MustRunAs   <no value>   false            ["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
hostnetwork-v2          false   ["NET_BIND_SERVICE"]    MustRunAs   MustRunAsRange     MustRunAs   MustRunAs   <no value>   false
```

```
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
node-exporter                true    <no value>          RunAsAny    RunAsAny         RunAsAny
RunAsAny    <no value>  false          ["*"]
nonroot                      false   <no value>          MustRunAs   MustRunAsNonRoot
RunAsAny    RunAsAny    <no value>  false
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
nonroot-v2                   false   ["NET_BIND_SERVICE"]  MustRunAs   MustRunAsNonRoot
RunAsAny    RunAsAny    <no value>  false
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
privileged                   true    ["*"]               RunAsAny    RunAsAny         RunAsAny
RunAsAny    <no value>  false          ["*"]
restricted                   false   <no value>          MustRunAs   MustRunAsRange
MustRunAs   RunAsAny    <no value>  false
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
restricted-v2                false   ["NET_BIND_SERVICE"]  MustRunAs   MustRunAsRange
MustRunAs   RunAsAny    <no value>  false
["configMap","downwardAPI","emptyDir","persistentVolumeClaim","projected","secret"]
```

12.7.2. 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:               <none> (2)
Settings:
  Allow Privileged:                 false
  Allow Privilege Escalation:       true
  Default Add Capabilities:         <none>
  Required Drop Capabilities:       KILL,MKNOD,SETUID,SETGID
  Allowed Capabilities:             <none>
  Allowed Seccomp Profiles:         <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,secret
  Allowed Flexvolumes:              <all>
  Allowed Unsafe Sysctls:           <none>
```

```
Forbidden Sysctls:                <none>
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。

12.8. 其他资源

- 获取支持

第 13 章 了解并管理 POD 安全准入

Pod 安全准入是 Kubernetes pod 安全标准的实现。使用 pod 安全准入来限制 pod 的行为。

13.1. 关于 POD 安全准入

Red Hat OpenShift Service on AWS 包括 Kubernetes pod 安全准入。不遵循全局或命名空间级别定义的 pod 安全准入的 Pod 不会被接受到集群且无法运行。

在全局范围内，会强制 privileged 配置集，restricted 配置集用于警告和审核。

您还可以在命名空间级别配置 pod 安全准入设置。

重要

不要在默认项目中运行工作负载或共享对默认项目的访问权限。为运行核心集群组件保留默认项目。

以下默认项目被视为具有高度特权：default, kube-public, kube-system, openshift, openshift-infra, openshift-node，其他系统创建的项目的标签 openshift.io/run-level 被设置为 0 或 1。依赖于准入插件（如 pod 安全准入、安全性上下文约束、集群资源配额和镜像引用解析）的功能无法在高特权项目中工作。

13.1.1. Pod 安全准入模式

您可以为命名空间配置以下 pod 安全准入模式：

表 13.1. Pod 安全准入模式

模式	标签	描述
enforce	pod-security.kubernetes.io/enforce	如果 pod 不符合集合配置集，则拒绝 pod 来自准入
audit	pod-security.kubernetes.io/audit	如果 pod 不符合集合配置集，日志审计事件

模式	标签	描述
warn	**pod-security.kubernetes.io/warn**	如果 pod 不符合集合配置集，则会显示警告

13.1.2. Pod 安全准入配置集

您可以将每个 pod 安全准入模式设置为以下配置集之一：

表 13.2. Pod 安全准入配置集

profile	描述
privileged	最低限制策略；允许已知特权升级
baseline	最低限制策略；防止已知特权升级
restricted	最严格的策略；遵循当前的 pod 强化最佳实践

13.1.3. 特权命名空间

以下系统命名空间总是设置为 privileged pod 安全准入配置集：

- *default*
- *kube-public*
- *kube-system*

您无法更改这些特权命名空间的 pod 安全配置集。

13.1.4. Pod 安全准入和安全性上下文约束

Pod 安全准入标准和安全上下文约束由两个独立的控制器协调并强制实施。这两个控制器使用以下进程独立工作，以强制执行安全策略：

1. 安全性上下文约束控制器可能会根据 pod 分配的 SCC 模拟一些安全上下文字段。例如，如果 seccomp 配置集为空或未设置，如果 pod 的分配的 SCC 将 seccompProfiles 字段强制为 runtime/default，控制器会将默认类型设置为 RuntimeDefault。

2. 安全性上下文约束控制器根据匹配的 SCC 验证 Pod 的安全上下文。

3. pod 安全准入控制器根据分配给命名空间的 pod 安全标准验证 pod 的安全上下文。

13.2. 关于 POD 安全准入同步

除了全局 pod 安全准入控制配置外，还存在一个控制器，它会根据给定命名空间中的服务帐户的 SCC 权限将 pod 安全准入控制 warn 和 audit 标签应用到命名空间。

控制器检查 ServiceAccount 对象权限，以便在每个命名空间中使用安全性上下文约束。安全性上下文约束 (SCC) 根据其字段值映射到 Pod 安全配置集，控制器使用这些翻译配置集。Pod 安全准入 warn 和 audit 标签被设置为命名空间中的最特权 pod 安全配置集，以防止在创建 pod 时显示警告和日志记录审计事件。

命名空间标签基于对命名空间本地服务帐户权限的考虑。

直接应用 pod 可能会使用运行 Pod 的用户的 SCC 特权。但是，在自动标记过程中不会考虑用户权限。

13.2.1. Pod 安全准入同步命名空间排除

在系统创建的命名空间中永久禁用 Pod 安全准入同步，openshift Block 前缀的命名空间。

定义为集群有效负载一部分的命名空间会永久禁用 pod 安全准入同步。以下命名空间被永久禁用：

- default
- kube-node-lease

- *kube-system*
- *kube-public*
- *openshift*
- *所有带有 openshift- 前缀的系统创建命名空间。*

13.3. 控制 POD 安全准入同步

您可以为大多数命名空间启用或禁用自动 pod 安全准入同步。

重要

您无法在系统创建的命名空间中启用 pod 安全准入同步。如需更多信息，请参阅 Pod 安全准入同步命名空间排除。

流程

- *对于您要配置的每个命名空间，为 security.openshift.io/scc.podSecurityLabelSync 标签设置一个值：*
  - *要在命名空间中禁用 pod 安全准入标签同步，将 security.openshift.io/scc.podSecurityLabelSync 标签的值设置为 false。*

    运行以下命令：

    ```
    $ oc label namespace <namespace>
    security.openshift.io/scc.podSecurityLabelSync=false
    ```

  - *要在命名空间中启用 pod 安全准入标签同步，请将 security.openshift.io/scc.podSecurityLabelSync 标签的值设置为 true。*

    运行以下命令：

    -

```
$ oc label namespace <namespace>
security.openshift.io/scc.podSecurityLabelSync=true
```

其他资源

- Pod 安全准入同步命名空间排除

13.4. 为命名空间配置 POD 安全准入

您可以在命名空间级别配置 pod 安全准入设置。对于命名空间中的每个 pod 安全准入模式，您可以设置要使用的 pod 安全准入配置集。

流程

- 对于您要在命名空间上设置的每个 pod 安全准入模式，请运行以下命令：

```
$ oc label namespace <namespace> \
  pod-security.kubernetes.io/<mode>=<profile> \
  --overwrite
```

(1) 将 <namespace> 设置为要配置的命名空间。

(2) 将 <mode> 设置为 enforce, warn, 或 audit。将 <profile> 设置为 restricted, baseline, 或 privileged。

13.5. 关于 POD 安全准入警报

当 Kubernetes API 服务器报告 pod 安全准入控制器的审计级别时，会触发 PodSecurityViolation 警报。此警报持续一天。

查看 Kubernetes API 服务器审计日志，以调查触发的警报。例如，如果将全局执行设置为 restricted pod 安全级别，工作负载可能会出现故障。

如需有关识别 Pod 安全准入违反审计事件的帮助，请参阅 Kubernetes 文档中的 Audit 注解。

13.6. 其他资源

- 查看审计日志
- 管理安全性上下文约束

第 14 章 同步 LDAP 组

作为具有 dedicated-admin 角色的管理员，您可以使用组来管理用户、更改其权限并增强协作。您的组织可能已创建了用户组，并将其存储在 LDAP 服务器中。Red Hat OpenShift Service on AWS 可以与内部 Red Hat OpenShift Service on AWS 记录同步这些 LDAP 记录，可让您在一个位置管理组。Red Hat OpenShift Service on AWS 目前支持使用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。

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 dedicated-admin 权限才能同步组。

14.1. 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 Red Hat OpenShift Service on AWS 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。如果需要升级权限才能检索同步操作的条目，Red Hat OpenShift Service on AWS 会使用它。

3

用于绑定的可选密码。如果需要升级权限才能检索同步操作的条目，Red Hat OpenShift Service on AWS 会使用它。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，Red Hat OpenShift Service on AWS 将使用 system-trusted roots。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 14.1. 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。

LDAP 搜索范围	描述
sub	考虑根部是为查询给定的基本 DN 的整个子树。

表 14.2. 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。
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 Red Hat OpenShift Service on AWS 组的名称映射到在 LDAP 服务器上查找组的唯一标识符。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 服务器上有没有用户定义的名称映射的组，同步过程中的默认行为是使用作为 Red Hat OpenShift Service on AWS 组名称指定的属性。

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
```

14.1.1. 关于 RFC 2307 配置文件

RFC 2307 模式要求您为用户和组条目提供 LDAP 查询定义，以及在 Red Hat OpenShift Service on AWS 记录中代表它们的属性。

为清晰起见，您在 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组的用户，并使用组的名称作为通用名称。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

在 Red Hat OpenShift Service on AWS 组记录中用作用户名称的属性。

14.1.2. 关于 Active Directory 配置文件

Active Directory 模式要求您为用户条目提供 LDAP 查询定义，以及在内部 Red Hat OpenShift Service on AWS 组记录中代表它们的属性。

为清晰起见，您在 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组的用户，但根据 LDAP 服务器上的组名称定义组名称。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

在 Red Hat OpenShift Service on AWS 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。

14.1.3. 关于增强 Active Directory 配置文件

增强 Active Directory 模式要求您为用户条目和组条目提供 LDAP 查询定义，以及在 Red Hat OpenShift Service on AWS 组记录中代表它们的属性。

为清晰起见，您在 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组的用户，并使用组的名称作为通用名称。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

在 Red Hat OpenShift Service on AWS 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。

14.2. 运行 LDAP 同步

创建了同步配置文件后，您可以开始同步。Red Hat OpenShift Service on AWS 允许管理员执行多个与同一服务器不同的同步类型。

14.2.1. 将 LDAP 服务器与 Red Hat OpenShift Service on AWS 同步

您可以将 LDAP 服务器上的所有组与 Red Hat OpenShift Service on AWS 同步。

先决条件

- 创建同步配置文件。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

流程

- 将 LDAP 服务器上的所有组与 Red Hat OpenShift Service on AWS 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，因此您必须在 oc adm groups sync 命令上设置 --confirm 标志，才能更改 Red Hat OpenShift Service on AWS 组记录。

14.2.2. 将 Red Hat OpenShift Service on AWS 组与 LDAP 服务器同步

您可以同步 Red Hat OpenShift Service on AWS 中已与配置文件中指定的 LDAP 服务器中的组对应的组。

先决条件

- 创建同步配置文件。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

流程

- 将 Red Hat OpenShift Service on AWS 组与 LDAP 服务器同步：

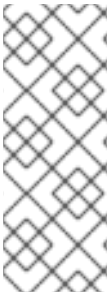
```
$ oc adm groups sync --type=openshift --sync-config=config.yaml --confirm
```

注意

默认情况下，所有组同步操作都是空运行，因此您必须在 oc adm groups sync 命令上设置 --confirm 标志，才能更改 Red Hat OpenShift Service on AWS 组记录。

14.2.3. 将 LDAP 服务器上的子组与 Red Hat OpenShift Service on AWS 同步

您可以使用白名单文件、黑名单文件或两者，将 LDAP 组的子集与 AWS 上的 Red Hat OpenShift Service 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。这些指南适用于在 LDAP 服务器上找到的组，以及 Red Hat OpenShift Service on AWS 中已存在的组。

先决条件

- *创建同步配置文件。*
- *您可以使用具有 dedicated-admin 角色的用户访问集群。*

流程

- *要将 LDAP 组的子集与 AWS 上的 Red Hat OpenShift Service 同步，请使用以下命令：*

```
$ 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，因此您必须在 oc adm groups sync 命令上设置 --confirm 标志，才能更改 Red Hat OpenShift Service on AWS 组记录。

14.3. 运行组修剪任务

如果创建它们的 LDAP 服务器上的记录不再存在，管理员也可以选择从 Red Hat OpenShift Service on AWS 记录中删除组。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
```

14.4. 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

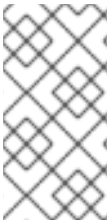
这些示例假定所有用户都是其各自组的直接成员。具体而言，没有任何组的成员是其他组。如需有关如何同步嵌套组的信息，请参见嵌套成员资格同步示例。

14.4.1. 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。

- 在 Red Hat OpenShift Service on AWS 中生成的组记录将在同步后进行。



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。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

流程

- 使用 rfc2307_config.yaml 文件运行同步：

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

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 rfc2307_config.yaml 文件创建的 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组最后一次与 LDAP 服务器同步的时间，格式为 ISO 6801。

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。

14.4.2. 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，则命名 Red Hat OpenShift Service on AWS 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。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

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
```

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 rfc2307_config_user_defined.yaml 文件创建的 Red Hat OpenShift Service on AWS 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。

14.4.3. 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。如果没有正确配置 LDAP 组同步，这可能会导致 AWS 组中的同步 Red Hat OpenShift Service 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

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。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

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
```

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 rfc2307_config.yaml 文件创建的 Red Hat OpenShift Service on AWS 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。

14.4.4. 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。

- 您可以使用具有 dedicated-admin 角色的用户访问集群。

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
```

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 active_directory_config.yaml 文件创建的 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组最后一次与 LDAP 服务器同步的时间，格式为 ISO 6801。

(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)

5

属于组的成员的用户，名称由同步文件指定。

14.4.5. 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。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

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
```

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 augmented_active_directory_config.yaml 文件创建的 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组最后一次与 LDAP 服务器同步的时间，格式为 ISO 6801。

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。

14.4.5.1. LDAP 嵌套成员资格同步示例

Red Hat OpenShift Service on AWS 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。
- 在 Red Hat OpenShift Service on AWS 中生成的组记录将在同步后进行。

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。

当嵌套组与 Active Directory 同步时，您必须为用户条目和组条目提供 LDAP 查询定义，以及在内部 Red Hat OpenShift Service on AWS 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。

为清晰起见，您在 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组的用户，并使用组的名称作为通用名称。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

在 Red Hat OpenShift Service on AWS 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。
- 您可以使用具有 dedicated-admin 角色的用户访问集群。

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。

Red Hat OpenShift Service on AWS 会创建以下组记录，因为上述同步操作的结果：

使用 augmented_active_directory_config_nested.yaml 文件创建的 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组最后一次与 LDAP 服务器同步的时间，格式为 ISO 6801。

(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。

14.5. 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。

14.5.1. 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

名称	描述	模式
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
groupUIDNameMapping	LDAP 组 UID 的可选直接映射到 Red Hat OpenShift Service on AWS 组名称。	object
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

14.5.2. 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

名称	描述	模式
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

14.5.3. 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

14.5.4. 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 Red Hat OpenShift Service on AWS 组的名称。	字符串数组
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	定义将按顺序使用 LDAP 用户条目上的哪些属性作为其 Red Hat OpenShift Service on AWS 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。在 Red Hat OpenShift Service on AWS 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

名称	描述	模式
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。如果为 **false**，如果用户查询没有找到任何信息，LDAP 同步任务将失败。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

14.5.5. 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 Red Hat OpenShift Service on AWS 用户名。在 Red Hat OpenShift Service on AWS 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

14.5.6. 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 Red Hat OpenShift Service on AWS 用户名。在 Red Hat OpenShift Service on AWS 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 Red Hat OpenShift Service on AWS 组的名称。	字符串数组