Red Hat OpenShift Service on AWS 4

認証および認可

Red Hat OpenShift Service on AWS クラスターのセキュリティー保護

Last Updated: 2024-04-13

Red Hat OpenShift Service on AWS 4 認証および認可

Red Hat OpenShift Service on AWS クラスターのセキュリティー保護

法律上の通知

概要

このドキュメントでは、Red Hat OpenShift Service on AWS (ROSA) クラスターのセキュリティー保護を説明します。

目次

第1章 認証および認可の概要 4
1.1. RED HAT OPENSHIFT SERVICE ON AWS の認証および認可に関する一般的な用語集 4
1.2. RED HAT OPENSHIFT SERVICE ON AWS での認証について 5
1.3. RED HAT OPENSHIFT SERVICE ON AWS での認可について 5

第2章 認証について 7
2.1. ユーザー 7
2.2. グループ 7
2.3. API 認証 8

第3章 ユーザーが所有する OAUTH アクセストークンの管理 10
3.1. ユーザーが所有する OAUTH アクセストークンのリスト表示 10
3.2. ユーザーが所有する OAUTH アクセストークンの詳細の表示 10
3.3. ユーザーが所有する OAUTH アクセストークンの削除 11

第4章 アイデンティティープロバイダーの設定 13
4.1. アイデンティティープロバイダーについて 13
4.2. GITHUB アイデンティティープロバイダーの設定 14
4.3. GITLAB アイデンティティープロバイダーの設定 16
4.4. GOOGLE アイデンティティープロバイダーの設定 17
4.5. LDAP アイデンティティープロバイダーの設定 18
4.6. OPENID アイデンティティープロバイダーの設定 20
4.7. HTPASSWD アイデンティティープロバイダーの設定 22
4.8. 関連情報 23

第5章 RBAC の使用によるパーミッションの定義および適用 24
5.1. RBAC の概要 24
5.2. プロジェクトおよび NAMESPACE 27
5.3. デフォルトプロジェクト 28
5.4. クラスターロールおよびバインディングの表示 29
5.5. ローカルのロールバインディングの表示 36
5.6. ロールのユーザーへの追加 37
5.7. ローカルロールの作成 39
5.8. ローカルロールバインディングのコマンド 40
5.9. クラスターのロールバインディングコマンド 41
5.10. CLUSTER-ADMIN アクセス権限の付与 41
5.11. DEDICATED-ADMIN アクセスの取り消し 42

第6章 サービスアカウントの概要および作成 44
6.1. サービスアカウントの概要 44
6.2. サービスアカウントの作成 44
6.3. ロールをサービスアカウントに付与する例 45

第7章 USING SERVICE ACCOUNTS IN APPLICATIONS 48
7.1. サービスアカウントの概要 48
7.2. デフォルトのサービスアカウント 48
7.3. サービスアカウントの作成 51

第8章 サービスアカウントの OAUTH クライアントとしての使用 53
8.1. OAUTH クライアントとしてのサービスアカウント 53

第9章 サービスアカウントの AWS IAM ロールの引き受け 56
9.1. サービスアカウントが SRE 所有のプロジェクトで AWS IAM ロールを引き受ける方法 56

9.2. サービスアカウントがユーザー定義プロジェクトで AWS IAM ロールを引き受ける方法 58
9.3. 独自の POD で AWS IAM ロールを引き受ける 60
9.4. 関連情報 70

第10章 スコープトークン **71**
10.1. トークンのスコープについて 71

第11章 バインドされたサービスアカウントトークンの使用 **72**
11.1. バインドされたサービスアカウントトークンについて 72
11.2. ボリュームローテーションを使用したバインドされたサービスアカウントトークンの設定 72
11.3. POD の外部でバインドされたサービスアカウントトークンを作成する 73

第12章 SECURITY CONTEXT CONSTRAINTS の管理 **75**
12.1. SECURITY CONTEXT CONSTRAINTS について 75
12.2. 事前に割り当てられる SECURITY CONTEXT CONSTRAINTS 値について 86
12.3. SECURITY CONTEXT CONSTRAINTS の例 87
12.4. セキュリティーコンテキスト制約の作成 89
12.5. 特定の SCC を必要とするワークロードの設定 91
12.6. SECURITY CONTEXT CONSTRAINTS へのロールベースのアクセス 92
12.7. SECURITY CONTEXT CONSTRAINTS コマンドのリファレンス 93
12.8. 関連情報 95

第13章 POD セキュリティーアドミッションの理解と管理 **96**
13.1. POD セキュリティーアドミッションについて 96
13.2. POD セキュリティーアドミッション同期について 97
13.3. POD セキュリティーアドミッションの同期制御 98
13.4. NAMESPACE の POD セキュリティーアドミッションの設定 99
13.5. POD セキュリティーアドミッションアラート 99
13.6. 関連情報 99

第14章 LDAP グループの同期 **100**
14.1. LDAP 同期の設定について 100
14.2. LDAP 同期の実行 105
14.3. グループのプルーニングジョブの実行 107
14.4. LDAP グループの同期の例 107
14.5. LDAP 同期設定の仕様 120

第1章 認証および認可の概要

1.1. RED HAT OPENSHIFT SERVICE ON AWS の認証および認可に関する一般的な用語集

この用語集では、Red Hat OpenShift Service on AWS の認証および認可で使用される一般的な用語を定義します。

認証

認証は、Red Hat OpenShift Service on AWS クラスターへのアクセスを決定し、認証されたユーザーのみが Red Hat OpenShift Service on AWS クラスターにアクセスできるようにします。

認可

認可は、要求されたアクションを実行する権限を識別されたユーザーが持っているかどうかを決定します。

ベアラートークン

ベアラートークンは、ヘッダー **Authorization: Bearer <token>** で API を認証するために使用されます。

ConfigMap

設定マップは、設定データを Pod に注入する方法を提供します。タイプ **ConfigMap** のボリューム内の設定マップに格納されたデータを参照できます。Pod で実行しているアプリケーションは、このデータを使用できます。

コンテナー

ソフトウェアとそのすべての依存関係を設定する軽量で実行可能なイメージ。コンテナーはオペレーティングシステムを仮想化するため、データセンター、パブリッククラウドまたはプライベートクラウド、またはローカルホストでコンテナーを実行できます。

カスタムリソース (CR)

CR は Kubernetes API のエクステンションです。

グループ

グループはユーザーの集まりです。グループは、一度に複数のユーザーに権限を付与する場合に便利です。

HTPasswd

HTPasswd は、HTTP ユーザーの認証用のユーザー名とパスワードを格納するファイルを更新します。

Keystone

Keystone は、ID、トークン、カタログ、およびポリシーサービスを提供する Red Hat OpenStack Platform (RHOSP) プロジェクトです。

Lightweight Directory Access Protocol (LDAP)

LDAP は、ユーザー情報を照会するプロトコルです。

namespace

namespace は、すべてのプロセスから見える特定のシステムリソースを分離します。namespace 内では、その namespace のメンバーであるプロセスのみがそれらのリソースを参照できます。

ノード

ノードは、Red Hat OpenShift Service on AWS クラスター内のワーカーマシンです。ノードは、仮想マシン (VM) または物理マシンのいずれかです。

OAuth クライアント

OAuth クライアントは、ベアラートークンを取得するために使用されます。

OAuth サーバー

Red Hat OpenShift Service on AWS コントロールプレーンには、設定されたアイデンティティープロバイダーからユーザーのアイデンティティーを決定し、アクセストークンを作成する組み込みの OAuth サーバーが含まれています。

OpenID Connect

OpenID Connect は、ユーザーが Single Sign-On (SSO) を使用して OpenID プロバイダーを使用するサイトにアクセスすることを認証するためのプロトコルです。

pod

Pod は、Kubernetes における最小の論理単位です。Pod には、ワーカーノードで実行される 1 つ以上のコンテナーが含まれます。

通常ユーザー

最初のログイン時または API 経由でクラスター内に自動的に作成されるユーザー。

リクエストヘッダー

要求ヘッダーは、サーバーが要求の応答を追跡できるように、HTTP 要求コンテキストに関する情報を提供するために使用される HTTP ヘッダーです。

ロールベースのアクセス制御 (RBAC)

クラスターユーザーとワークロードが、ロールを実行するために必要なリソースにのみアクセスできるようにするための重要なセキュリティーコントロール。

サービスアカウント

サービスアカウントは、クラスターコンポーネントまたはアプリケーションによって使用されます。

システムユーザー

クラスターのインストール時に自動的に作成されるユーザー。

ユーザー

ユーザーは、API に要求を送信できるエンティティーです。

1.2. RED HAT OPENSHIFT SERVICE ON AWS での認証について

Red Hat OpenShift Service on AWS へのアクセスを制御する際には、**dedicated-admin** ロールを持つ管理者が ユーザー認証 を設定し、承認されたユーザーにのみクラスターへのアクセス権を付与します。

ユーザーが Red Hat OpenShift Service on AWS と対話するには、まず何らかの方法で Red Hat OpenShift Service on AWS API に対して認証する必要があります。Red Hat OpenShift Service on AWS API への要求で、OAuth アクセストークンまたは X.509 クライアント証明書 を提供することで認証できます。

注記

有効なアクセストークンまたは証明書を提示しない場合、要求は認証されず、HTTP 401 エラーが発生します。

管理者は、アイデンティティプロバイダーを設定することで認証を設定できます。Red Hat OpenShift Service on AWS でサポートされているアイデンティティプロバイダー を定義し、クラスターに追加できます。

1.3. RED HAT OPENSHIFT SERVICE ON AWS での認可について

認可は、要求されたアクションを実行する権限を識別されたユーザーが持っているかどうかを決定することです。

管理者は、権限を定義し、ルール、ロール、バインディングなどの RBAC オブジェクト を使用してそれらをユーザーに割り当てることができます。Red Hat OpenShift Service on AWS での認可の仕組みを理解するには、認可の評価 を参照してください。

プロジェクトと namespace を介して、Red Hat OpenShift Service on AWS クラスターへのアクセスを制御することもできます。

クラスターへのユーザーアクセスを制御するだけでなく、セキュリティーコンテキスト制約 (SCC) を使用して、Pod が実行できるアクションとアクセスできるリソースを制御することもできます。

次のタスクを通じて、Red Hat OpenShift Service on AWS の認証を管理できます。

- ローカルおよびクラスターのロールとバインディングの表示。
- ローカルロールを作成し、それをユーザーまたはグループに割り当てます。
- ユーザーまたはグループへのクラスターロールの割り当て: Red Hat OpenShift Service on AWS には、デフォルトのクラスターロール のセットがあります。これらをユーザーまたはグループに追加 できます。
- クラスター管理者および dedicated-admin ユーザーの作成: Red Hat OpenShift Service on AWS クラスターを作成したユーザーは、他の **cluster-admin** および **dedicated-admin** ユーザーにアクセスを許可できます。
- サービスアカウントの作成: サービスアカウントは、通常のユーザークレデンシャルを共有せずに API アクセスを制御する柔軟な方法を提供します。ユーザーは、アプリケーションでサービスアカウントを作成して使用したり、OAuth クライアント として使用したりできます。
- スコープトークン: スコープトークンは、特定の操作のみを実行できる特定のユーザーとして識別するトークンです。スコープ付きトークンを作成して、パーミッションの一部を別のユーザーまたはサービスアカウントに委任できます。
- LDAP グループの同期: LDAP サーバーに保存されているグループを Red Hat OpenShift Service on AWS ユーザーグループと同期 することにより、ユーザーグループを 1 カ所で管理できます。

第2章 認証について

ユーザーが Red Hat OpenShift Service on AWS と対話するには、まずユーザークラスターに対して認証する必要があります。認証層は、Red Hat OpenShift Service on AWS API への要求に関連するユーザーを識別します。その後、認可層が要求元のユーザーに関する情報を使用して、要求を許可するかどうかを決定します。

2.1. ユーザー

Red Hat OpenShift Service on AWS の **ユーザー** は、Red Hat OpenShift Service on AWS API への要求を実行できるエンティティーです。Red Hat OpenShift Service on AWS の **User** オブジェクトは、それらおよびそれらのグループにロールを追加してシステム内の権限を付与できるアクターを表します。通常、このオブジェクトは Red Hat OpenShift Service on AWS と対話する開発者または管理者のアカウントを表します。

ユーザーにはいくつかのタイプが存在します。

ユーザータイプ	説明
Regular users	これは、大半の対話型の Red Hat OpenShift Service on AWS ユーザーが表現される方法です。通常ユーザーは、初回ログイン時にシステムに自動的に作成され、API で作成できます。通常ユーザーは **User** オブジェクトで表されます。例: **joe alice**
System users	これらの多くは、インフラストラクチャーが API と安全に対話できるようにすることを主な目的として定義される際に自動的に作成されます。これらには、クラスター管理者 (すべてのものへのアクセスを持つ)、ノードごとのユーザー、ルーターおよびレジストリーで使用できるユーザー、その他が含まれます。最後に、非認証要求に対してデフォルトで使用される **anonymous** システムユーザーもあります。例: **system:admin system:openshift-registry system:node:node1.example.com**
Service accounts	プロジェクトに関連付けられる特殊なシステムユーザーがあります。それらの中には、プロジェクトの初回作成時に自動作成されるものもあれば、プロジェクト管理者が各プロジェクトのコンテンツへのアクセスを定義するために追加で作成するものもあります。サービスアカウントは **ServiceAccount** オブジェクトで表されます。例: **system:serviceaccount:default:deployer** **system:serviceaccount:foo:builder**

それぞれのユーザーには、Red Hat OpenShift Service on AWS にアクセスするために何らかの認証が必要になります。認証がないか、認証が無効の API 要求は、**anonymous** システムユーザーによる要求として認証されます。認証が実行されると、認可されているユーザーの実行内容がポリシーによって決定されます。

2.2. グループ

ユーザーは 1 つ以上の **グループ** に割り当てることができます。それぞれのグループはユーザーの特定のセットを表します。グループは、認可ポリシーを管理し、個々のユーザーにではなく、一度に複数ユーザーにパーミッションを付与する場合などに役立ちます。 たとえば、アクセスをユーザーに個別に付与するのではなく、プロジェクト内の複数のオブジェクトに対するアクセスを許可できます。

明示的に定義されるグループのほかにも、システムグループまたは **仮想グループ** がクラスターによって自動的にプロビジョニングされます。

以下のデフォルト仮想グループは最も重要なグループになります。

仮想グループ	説明
system:authenticated	認証されたユーザーに自動的に関連付けられます。
system:authenticated:oauth	OAuth アクセストークンで認証されたすべてのユーザーに自動的に関連付けられます。
system:unauthenticated	認証されていないすべてのユーザーに自動的に関連付けられます。

2.3. API 認証

Red Hat OpenShift Service on AWS API への要求は以下の方法で認証されます。

OAuth アクセストークン

- **<namespace_route>/oauth/authorize** および **<namespace_route>/oauth/token** エンドポイントを使用して Red Hat OpenShift Service on AWS OAuth サーバーから取得されます。
- **Authorization: Bearer…** ヘッダーとして送信されます。
- websocket 要求の **base64url.bearer.authorization.k8s.io.<base64url-encoded-token>** 形式の websocket サブプロトコルヘッダーとして送信されます。

X.509 クライアント証明書

- API サーバーへの HTTPS 接続を要求します。
- 信頼される認証局バンドルに対して API サーバーによって検証されます。
- API サーバーは証明書を作成し、これをコントローラーに配布してそれらを認証できるようにします。

無効なアクセストークンまたは無効な証明書での要求は認証層によって拒否され、**401** エラーが出されます。

アクセストークンまたは証明書が提供されない場合、認証層は **system:anonymous** 仮想ユーザーおよび **system:unauthenticated** 仮想グループを要求に割り当てます。これにより、認可層は匿名ユーザーが実行できる要求 (ある場合) を決定できます。

2.3.1. Red Hat OpenShift Service on AWS OAuth サーバー

Red Hat OpenShift Service on AWS のマスターには、OAuth サーバーが組み込まれています。ユーザーは OAuth アクセストークンを取得して、API に対して認証します。

新しい OAuth のトークンが要求されると、OAuth サーバーは設定済みのアイデンティティプロバイダーを使用して要求したユーザーのアイデンティティーを判別します。

次に、そのアイデンティティーがマップするユーザーを判別し、そのユーザーのアクセストークンを作成し、そのトークンを使用できるように返します。

2.3.1.1. OAuth トークン要求

OAuth トークンのすべての要求は、トークンを受信し、使用する OAuth クライアントを指定する必要があります。Red Hat OpenShift Service on AWS API の起動時に、次の OAuth クライアントが自動的に作成されます。

OAuth クライアント	使用法
openshift-browser-client	対話式ログインを処理できるユーザーエージェントで **<namespace_route>/oauth/token/request** でトークンを要求します。[1]
openshift-challenging-client	**WWW-Authenticate** チャレンジを処理できるユーザーエージェントでトークンを要求します。

1. **<namespace_route>** は namespace のルートを参照します。これは、以下のコマンドを実行して確認できます。

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

OAuth トークンのすべての要求には **<namespace_route>/oauth/authorize** への要求が必要になります。ほとんどの認証統合では、このエンドポイントの前に認証プロキシーを配置するか、サポートするアイデンティティープロバイダーに対して認証情報を検証するように Red Hat OpenShift Service on AWS を設定します。**<namespace_route>/oauth/authorize** の要求は、CLI などの対話式ログインページを表示できないユーザーエージェントから送られる場合があります。そのため、Red Hat OpenShift Service on AWS は、対話式ログインフローのほかにも **WWW-Authenticate** チャレンジを使用した認証をサポートします。

認証プロキシーが **<namespace_route>/oauth/authorize** エンドポイントの前に配置される場合、対話式ログインページを表示したり、対話式ログインフローにリダイレクトする代わりに、認証されていない、ブラウザー以外のユーザーエージェントの **WWW-Authenticate** チャレンジを送信します。

注記

ブラウザークライアントに対するクロスサイトリクエストフォージェリー (CSRF) 攻撃を防止するため、基本的な認証チャレンジは **X-CSRF-Token** ヘッダーが要求に存在する場合にのみ送信されます。基本的な **WWW-Authenticate** チャレンジを受信する必要があるクライアントでは、このヘッダーを空でない値に設定する必要があります。

認証プロキシーが **WWW-Authenticate** チャレンジをサポートしないか、Red Hat OpenShift Service on AWS が WWW-Authenticate チャレンジをサポートしないアイデンティティープロバイダーを使用するように設定されている場合、ユーザーはブラウザーで **<namespace_route>/oauth/token/request** からトークンを手動で取得する必要があります。

第3章 ユーザーが所有する OAUTH アクセストークンの管理

ユーザーは、独自の OAuth アクセストークンを確認し、不要になったものを削除できます。

3.1. ユーザーが所有する OAUTH アクセストークンのリスト表示

ユーザーが所有する OAuth アクセストークンをリスト表示できます。トークン名には機密性がなく、ログインには使用できません。

手順

- ユーザーが所有する OAuth アクセストークンをリスト表示します。

```
$ oc get useroauthaccesstokens
```

出力例

```
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
```

- 特定の OAuth クライアントのユーザーが所有する OAuth アクセストークンをリスト表示します。

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

出力例

```
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. ユーザーが所有する OAUTH アクセストークンの詳細の表示

ユーザーが所有する OAuth アクセストークンの詳細を表示します。

手順

- ユーザーが所有する OAuth アクセストークンの詳細を記述します。

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

出力例

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

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

1. トークンの sha256 ハッシュであるトークン名。トークン名には機密性がなく、ログインには使用できません。
2. トークンの発信元の場所を記述するクライアント名。
3. このトークンが期限切れになるまでの時間 (秒単位)。
4. OAuth サーバーにトークンの非アクティブタイムアウトが設定されている場合、これは、作成された時間からこのトークンが使用されなくなるまでの時間 (秒単位) になります。
5. このトークンのスコープ。
6. このトークンに関連付けられたユーザー名。

3.3. ユーザーが所有する OAUTH アクセストークンの削除

oc logout コマンドは、アクティブなセッションの OAuth トークンのみを無効にします。以下の手順を使用して、不要になったユーザーが所有する OAuth トークンを削除できます。

OAuth アクセストークンを削除すると、そのトークンを使用するすべてのセッションからユーザーをログアウトします。

手順

- ユーザーが所有する OAuth アクセストークンを削除します。

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

出力例

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

第4章 アイデンティティープロバイダーの設定

Red Hat OpenShift Service on AWS クラスターを作成したら、アイデンティティプロバイダーを設定して、ユーザーがクラスターにログインしてアクセスする方法を決定する必要があります。

以下のトピックでは、OpenShift Cluster Manager コンソールを使用してアイデンティティープロバイダーを設定する方法を説明します。または、ROSA CLI (**rosa**) を使用してアイデンティティープロバイダーを設定し、クラスターにアクセスできます。

4.1. アイデンティティープロバイダーについて

Red Hat OpenShift Service on AWS には、ビルトイン OAuth サーバーが含まれます。開発者および管理者は OAuth アクセストークンを取得して、API に対して認証します。管理者は、クラスターのインストール後に、OAuth をアイデンティティープロバイダーを指定するように設定できます。アイデンティティープロバイダーを設定すると、ユーザーはログインし、クラスターにアクセスできます。

4.1.1. サポートされるアイデンティティープロバイダー

以下の種類のアイデンティティープロバイダーを設定できます。

アイデンティティープロバイダー	説明
GitHub または GitHub Enterprise	GitHub または GitHub Enterprise の OAuth 認証サーバーに対して、ユーザー名とパスワードを検証するように Github アイデンティティープロバイダーを設定します。
GitLab	GitLab.com またはその他の GitLab インスタンスをアイデンティティープロバイダーとして使用するように GitLab アイデンティティープロバイダーを設定します。
Google	Google の OpenID Connect 統合機能を使用して Google アイデンティティープロバイダーを設定します。
LDAP	単純なバインド認証を使用して、LDAPv3 サーバーに対してユーザー名とパスワードを検証するように LDAP アイデンティティープロバイダーを設定します。
OpenID Connect	Authorization Code Flow を使用して OpenID Connect アイデンティティープロバイダーと統合するように OpenID Connect (OIDC) アイデンティティープロバイダーを設定します。
htpasswd	単一の静的管理ユーザー用に htpasswd アイデンティティープロバイダーを設定します。問題のトラブルシューティングを行うには、ユーザーとしてクラスターにログインできます。 **重要** htpasswd ID プロバイダーオプションは、単一の静的管理ユーザーを作成できるようにするためだけに含まれています。htpasswd は、Red Hat OpenShift Service on AWS の汎用 ID プロバイダーとしてはサポートされていません。単一ユーザーを設定する手順は、**htpasswd アイデンティティープロバイダーの設定** を参照してください。

4.1.2. アイデンティティープロバイダーパラメーター

以下のパラメーターは、すべてのアイデンティティープロバイダーに共通するパラメーターです。

パラメーター	説明
name	プロバイダー名は、プロバイダーのユーザー名に接頭辞として付加され、アイデンティティー名が作成されます。
mappingMethod	新規アイデンティティーがログイン時にユーザーにマップされる方法を定義します。以下の値のいずれかを入力します。 **claim** デフォルトの値です。アイデンティティーの推奨ユーザー名を持つユーザーをプロビジョニングします。そのユーザー名を持つユーザーがすでに別のアイデンティティーにマッピングされている場合は失敗します。 **lookup** 既存のアイデンティティー、ユーザーアイデンティティーマッピング、およびユーザーを検索しますが、ユーザーまたはアイデンティティーの自動プロビジョニングは行いません。これにより、クラスター管理者は手動で、または外部のプロセスを使用してアイデンティティーとユーザーを設定できます。この方法を使用する場合は、ユーザーを手動でプロビジョニングする必要があります。 **add** アイデンティティーの推奨ユーザー名を持つユーザーをプロビジョニングします。推奨ユーザー名を持つユーザーがすでに存在する場合、アイデンティティーは既存のユーザーにマッピングされ、そのユーザーの既存のアイデンティティーマッピングに追加されます。これは、同じユーザーセットを識別して同じユーザー名にマッピングするアイデンティティープロバイダーが複数設定されている場合に必要です。

注記

mappingMethod パラメーターを **add** に設定すると、アイデンティティープロバイダーの追加または変更時に新規プロバイダーのアイデンティティーを既存ユーザーにマッピングできます。

4.2. GITHUB アイデンティティープロバイダーの設定

GitHub または GitHub Enterprise の OAuth 認証サーバーに対してユーザー名とパスワードを検証し、Red Hat OpenShift Service on AWS クラスターにアクセスするように GitHub アイデンティティープロバイダーを設定します。OAuth は Red Hat OpenShift Service on AWS と GitHub または GitHub Enterprise 間のトークン交換フローを容易にします。

警告

GitHub 認証を設定することによって、ユーザーは GitHub 認証情報を使用して Red Hat OpenShift Service on AWS にログインできます。GitHub ユーザー ID を持つすべてのユーザーが Red Hat OpenShift Service on AWS クラスターにログインできないようにするために、アクセスを特定の GitHub 組織のユーザーに制限する必要があります。

前提条件

- OAuth アプリケーションを、GitHub 組織管理者によって GitHub 組織設定 内に直接作成している。
- GitHub 組織またはチーム が GitHub アカウントに設定されている。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、アイデンティティープロバイダーを設定する必要のあるクラスターを選択します。
2. **Access control** タブをクリックします。
3. **Add identity provider** をクリックします。

注記

クラスターの作成後に表示される警告メッセージの **Add Oauth configuration** リンクをクリックして、アイデンティティープロバイダーを設定することもできます。

4. ドロップダウンメニューから **GitHub** を選択します。
5. アイデンティティープロバイダーの一意の名前を入力します。この名前は後で変更することができません。
   - **OAuth callback URL** が指定のフィールドに自動的に生成されます。これを使用して GitHub アプリケーションを登録します。

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

     以下に例を示します。

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

6. アプリケーションを GitHub に登録します。
7. Red Hat OpenShift Service on AWS に戻り、ドロップダウンメニューからマッピング方法を選択します。ほとんどの場合は、**Claim** の使用が推奨されます。

8. GitHub から提供される **Client ID** および **Client secret** を入力します。
9. **hostname** を入力します。GitHub Enterprise のホステッドインスタンスを使用する場合は、ホスト名を入力する必要があります。
10. オプション: 認証局 (CA) ファイルを使用して、設定された GitHub Enterprise URL のサーバー証明書を検証できます。**Browse** をクリックして **CA ファイル** を見つけ、これをアイデンティティープロバイダーに割り当てます。
11. **Use organizations** または **Use teams** を選択し、アクセスを特定の GitHub 組織または GitHub チームに制限します。
12. アクセスを制限する組織またはチームの名前を入力します。**Add more** をクリックして、ユーザーが所属できる複数の組織またはチームを指定します。
13. **Confirm** をクリックします。

検証

- 設定されたアイデンティティープロバイダーが **Clusters** ページの **Access control** タブに表示されるようになりました。

4.3. GITLAB アイデンティティープロバイダーの設定

GitLab.com またはその他の GitLab インスタンスをアイデンティティープロバイダーとして使用するように GitLab アイデンティティープロバイダーを設定します。

前提条件

- GitLab バージョン 7.7.0 から 11.0 を使用する場合は、OAuth 統合 を使用して接続します。GitLab バージョン 11.1 以降の場合は、OAuth ではなく OpenID Connect (OIDC) を使用して接続します。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、アイデンティティープロバイダーを設定する必要のあるクラスターを選択します。
2. **Access control** タブをクリックします。
3. **Add identity provider** をクリックします。

注記

クラスターの作成後に表示される警告メッセージの **Add Oauth configuration** リンクをクリックして、アイデンティティープロバイダーを設定することもできます。

4. ドロップダウンメニューから **GitLab** を選択します。
5. アイデンティティープロバイダーの一意の名前を入力します。この名前は後で変更することができません。
    - **OAuth callback URL** が指定のフィールドに自動的に生成されます。この URL を GitLab に指定します。

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

以下に例を示します。

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

6. GitLab に新規アプリケーションを追加します。
7. Red Hat OpenShift Service on AWS に戻り、ドロップダウンメニューからマッピング方法を選択します。ほとんどの場合は、**Claim** の使用が推奨されます。
8. GitLab から提供される **Client ID** および **Client secret** を入力します。
9. GitLab プロバイダーの **URL** を入力します。
10. オプション: 認証局 (CA) ファイルを使用して、設定された GitLab URL のサーバー証明書を検証できます。**Browse** をクリックして **CA ファイル** を見つけ、これをアイデンティティープロバイダーに割り当てます。
11. **Confirm** をクリックします。

検証

- 設定されたアイデンティティープロバイダーが **Clusters** ページの **Access control** タブに表示されるようになりました。

4.4. GOOGLE アイデンティティープロバイダーの設定

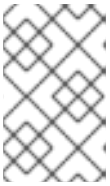
ユーザーが Google 認証情報で認証できるように Google アイデンティティープロバイダーを設定します。

警告

Google をアイデンティティープロバイダーとして使用することで、Google ユーザーはサーバーに対して認証されます。**hostedDomain** 設定属性を使用して、特定のホストドメインのメンバーに認証を限定することができます。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、アイデンティティープロバイダーを設定する必要のあるクラスターを選択します。
2. **Access control** タブをクリックします。
3. **Add identity provider** をクリックします。



注記

クラスターの作成後に表示される警告メッセージの **Add Oauth configuration** リンクをクリックして、アイデンティティープロバイダーを設定することもできます。

4. ドロップダウンメニューから **Google** を選択します。
5. アイデンティティープロバイダーの一意の名前を入力します。この名前は後で変更することができません。
   - **OAuth callback URL** が指定のフィールドに自動的に生成されます。この URL を Google に指定します。

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

     以下に例を示します。

     ```
     https://oauth-openshift.apps.openshift-cluster.example.com/oauth2callback/google
     ```
6. Google の OpenID Connect 統合機能 を使用して Google アイデンティティープロバイダーを設定します。
7. Red Hat OpenShift Service on AWS に戻り、ドロップダウンメニューからマッピング方法を選択します。ほとんどの場合は、**Claim** の使用が推奨されます。
8. 登録済みの Google プロジェクトの **Client ID** と、Google が発行する **Client secret** を入力します。
9. ホストされたドメインを入力して、ユーザーを Google Apps ドメインに制限します。
10. **Confirm** をクリックします。

検証

- 設定されたアイデンティティープロバイダーが **Clusters** ページの **Access control** タブに表示されるようになりました。

4.5. LDAP アイデンティティープロバイダーの設定

単純なバインド認証を使用して LDAPv3 サーバーに対してユーザー名とパスワードを検証するように LDAP アイデンティティープロバイダーを設定します。

前提条件

- LDAP アイデンティティープロバイダーを設定する場合は、設定済みの **LDAP URL** を入力する必要があります。設定される URL は、LDAP ホストと使用する検索パラメーターを指定する RFC 2255 URL です。URL の構文は以下のようになります。

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

URL コンポーネント	説明
ldap	通常の LDAP の場合は、文字列 **ldap** を使用します。セキュアな LDAP (LDAPS) の場合は、代わりに **ldaps** を使用します。
host:port	LDAP サーバーの名前とポートです。デフォルトは、ldap の場合は **localhost:389**、LDAPS の場合は **localhost:636** です。
basedn	すべての検索が開始されるディレクトリーのブランチの DN です。これは少なくともディレクトリーツリーの最上位になければなりませんが、ディレクトリーのサブツリーを指定することもできます。
attribute	検索対象の属性です。RFC 2255 はコンマ区切りの属性のリストを許可しますが、属性をどれだけ指定しても最初の属性のみが使用されます。属性を指定しない場合は、デフォルトで **uid** が使用されます。使用しているサブツリーのすべてのエントリー間で一意の属性を選択することを推奨します。
scope	検索の範囲です。**one** または **sub** のいずれかを指定できます。範囲を指定しない場合は、デフォルトの範囲として **sub** が使用されます。
filter	有効な LDAP 検索フィルターです。指定しない場合、デフォルトは **(objectClass=*)** です。

検索の実行時に属性、フィルター、指定したユーザー名が組み合わされて以下のような検索フィルターが作成されます。

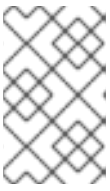
```
(&(<filter>)(<attribute>=<username>))
```

重要

LDAP ディレクトリーの検索に認証が必要な場合は、エントリー検索の実行に使用する **bindDN** と **bindPassword** を指定します。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、アイデンティティープロバイダーを設定する必要のあるクラスターを選択します。
2. **Access control** タブをクリックします。
3. **Add identity provider** をクリックします。

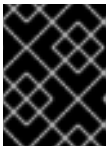


注記

クラスターの作成後に表示される警告メッセージの **Add Oauth configuration** リンクをクリックして、アイデンティティープロバイダーを設定することもできます。

4. ドロップダウンメニューから **LDAP** を選択します。

5. アイデンティティープロバイダーの一意の名前を入力します。この名前は後で変更することができません。
6. ドロップダウンメニューからマッピング方法を選択します。ほとんどの場合は、**Claim** の使用が推奨されます。
7. **LDAP URL** を入力して、使用する LDAP 検索パラメーターを指定します。
8. オプション: **Bind DN** および **Bind password** を入力します。
9. LDAP 属性をアイデンティティーにマップする属性を入力します。
   - 値をユーザー ID として使用する **ID** 属性を入力します。**Add more** をクリックして、複数の ID 属性を追加します。
   - オプション: 表示名の値として使用する **Preferred username** 属性を入力します。**Add more** をクリックして、優先する複数のユーザー名属性を追加します。
   - オプション: メールアドレスの値として使用する **Email** 属性を入力します。**Add more** をクリックして、複数のメール属性を追加します。
10. オプション: **Show advanced Options** をクリックし、認証局 (CA) ファイルを LDAP アイデンティティープロバイダーに追加し、設定された URL のサーバー証明書を検証します。**Browse** をクリックして **CA ファイル** を見つけ、これをアイデンティティープロバイダーに割り当てます。
11. オプション: 高度なオプションで、LDAP プロバイダーを **非セキュア** にするよう選択できます。このオプションを選択すると、CA ファイルは使用できません。



重要

非セキュアな LDAP 接続 (ldap:// またはポート 389) を使用している場合は、設定ウィザードで **で Insecure** オプションを確認する必要があります。

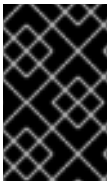
12. **Confirm** をクリックします。

検証

- 設定されたアイデンティティープロバイダーが **Clusters** ページの **Access control** タブに表示されるようになりました。

4.6. OPENID アイデンティティープロバイダーの設定

Authorization Code Flow を使用して OpenID Connect アイデンティティープロバイダーに統合するように OpenID アイデンティティープロバイダーを設定します。



重要

Red Hat OpenShift Service on AWS の認証 Operator では、設定済みの OpenID Connect アイデンティティープロバイダーが OpenID Connect Discovery 仕様を実装する必要があります。

要求は、OpenID アイデンティティープロバイダーから返される JWT **id_token** から読み取られ、指定される場合は発行者 URL によって返される JSON から読み取られます。

1つ以上の要求をユーザーのアイデンティティーを使用するように設定される必要があります。

また、どの要求をユーザーの推奨ユーザー名、表示名およびメールアドレスとして使用するか指定することができます。複数の要求が指定されている場合は、値が入力されている最初の要求が使用されます。標準の要求は以下の通りです。

要求	説明
preferred_username	ユーザーのプロビジョニング時に優先されるユーザー名です。**janedoe** などのユーザーを参照する際に使用する省略形の名前です。通常は、ユーザー名またはメールなどの、認証システムのユーザーのログインまたはユーザー名に対応する値です。
email	メールアドレス。
name	表示名。

詳細は、OpenID claim のドキュメント を参照してください。

前提条件

- OpenID Connect を設定する前に、Red Hat OpenShift Service on AWS クラスターで使用する Red Hat 製品またはサービスのインストール前提条件を確認してください。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、アイデンティティープロバイダーを設定する必要のあるクラスターを選択します。
2. **Access control** タブをクリックします。
3. **Add identity provider** をクリックします。

注記

クラスターの作成後に表示される警告メッセージの **Add Oauth configuration** リンクをクリックして、アイデンティティープロバイダーを設定することもできます。

4. ドロップダウンメニューから **OpenID** を選択します。
5. アイデンティティープロバイダーの一意の名前を入力します。この名前は後で変更することができません。
   - **OAuth callback URL** が指定のフィールドに自動的に生成されます。

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

     以下に例を示します。

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

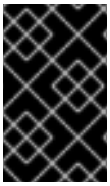
6. 認可リクエストを作成する 手順に従って、新しい OpenID Connect クライアントを OpenID ID プロバイダーに登録します。
7. Red Hat OpenShift Service on AWS に戻り、ドロップダウンメニューからマッピング方法を選択します。ほとんどの場合は、**Claim** の使用が推奨されます。
8. OpenID から提供される **Client ID** および **Client secret** を入力します。
9. **Issuer URL** を入力します。これは、OpenID プロバイダーが発行者 ID としてアサートする URL です。URL クエリーパラメーターまたはフラグメントのない https スキームを使用する必要があります。
10. メールアドレスの値として使用する **Email** 属性を入力します。**Add more** をクリックして、複数のメール属性を追加します。
11. 優先するユーザー名の値として使用する **Name** 属性を入力します。**Add more** をクリックして、優先する複数のユーザー名を追加します。
12. 表示名の値として使用する **Preferred username** 属性を入力します。**Add more** をクリックして、複数の表示名を追加します。
13. オプション: **Show advanced Options** をクリックし、認証局 (CA) ファイルを OpenID アイデンティティープロバイダーに追加します。
14. オプション: 高度なオプションから、**追加のスコープ** を追加できます。デフォルトでは、**OpenID** の範囲が要求されます。
15. **Confirm** をクリックします。

検証

- 設定されたアイデンティティープロバイダーが **Clusters** ページの **Access control** タブに表示されるようになりました。

4.7. HTPASSWD アイデンティティープロバイダーの設定

クラスター管理者権限で単一の静的ユーザーを作成するように htpasswd アイデンティティープロバイダーを設定します。問題のトラブルシューティングを行うには、ユーザーとしてクラスターにログインできます。



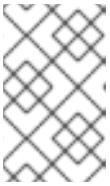
重要

htpasswd ID プロバイダーオプションは、単一の静的管理ユーザーを作成できるようにするためだけに含まれています。htpasswd は、Red Hat OpenShift Service on AWS の汎用 ID プロバイダーとしてはサポートされていません。

手順

1. OpenShift Cluster Manager から、**Clusters** ページに移動し、クラスターを選択します。
2. **Access control → Identity providers** の順に選択します。
3. **Add identity provider** をクリックします。
4. **Identity Provider** ドロップダウンメニューから **HTPasswd** を選択します。

5. アイデンティティープロバイダーの **Name** フィールドに一意の名前を追加します。
6. 静的ユーザーに推奨されるユーザー名およびパスワードを使用するか、独自のユーザー名およびパスワードを作成します。



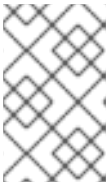
注記

この手順で定義した認証情報は、以下の手順で **Add** を選択した後に表示されません。認証情報を失った場合は、アイデンティティープロバイダーを再作成し、認証情報を再度定義する必要があります。

7. **Add** を選択して htpasswd アイデンティティープロバイダーおよび単一の静的ユーザーを作成します。
8. クラスターを管理する静的ユーザーにパーミッションを付与します。
   a. **Access control → Cluster Roles and Access** で、**Add user** を選択します。
   b. 前のステップで作成した静的ユーザーの **User ID** を入力します。
   c. **Add user** を選択して、管理者権限をユーザーに付与します。

検証

- 設定された htpasswd アイデンティティープロバイダーは、**Access control → Identity providers** ページに表示されます。



注記

アイデンティティープロバイダーの作成後に、同期は通常 2 分以内に完了します。htpasswd アイデンティティープロバイダーが利用可能になると、ユーザーとしてクラスターにログインできます。

- 管理ユーザーは、**Access control → Cluster Roles and Access** ページで確認できます。ユーザーの管理グループメンバーシップも表示されます。

4.8. 関連情報

- クラスターへのアクセス
- STS を使用する ROSA のデプロイメントワークフローについて

第5章 RBAC の使用によるパーミッションの定義および適用

5.1. RBAC の概要

Role-based Access Control (RBAC: ロールベースアクセス制御) オブジェクトは、ユーザーがプロジェクト内で所定のアクションを実行することが許可されるかどうかを決定します。

dedicated-admin ロールを持つ管理者は、クラスターロールおよびバインディングを使用して、Red Hat OpenShift Service on AWS プラットフォーム自体およびすべてのプロジェクトへの各種のアクセスレベルを持つユーザーを制御できます。

開発者はローカルロールとバインディングを使用して、プロジェクトにアクセスできるユーザーを制御できます。認可は認証とは異なる手順であることに注意してください。認証はアクションを実行するユーザーのアイデンティティーの判別により密接に関連しています。

認可は以下を使用して管理されます。

認可オブジェクト	説明
ルール	オブジェクトのセットで許可されている動詞のセット(例: ユーザーまたはサービスアカウントが Pod の **create** を実行できるかどうか)
ロール	ルールのコレクション。ユーザーおよびグループを複数のロールに関連付けたり、バインドしたりできます。
バインディング	ロールを使用したユーザー/グループ間の関連付けです。

2 つのレベルの RBAC ロールおよびバインディングが認可を制御します。

RBAC レベル	説明
クラスター RBAC	すべてのプロジェクトで適用可能なロールおよびバインディングです。**クラスターロール** はクラスター全体で存在し、**クラスターロールのバインディング** はクラスターロールのみを参照できます。
ローカル RBAC	所定のプロジェクトにスコープ設定されているロールおよびバインディングです。**ローカルロール** は単一プロジェクトのみに存在し、ローカルロールのバインディングはクラスターロールおよびローカルロールの **両方** を参照できます。

クラスターのロールバインディングは、クラスターレベルで存在するバインディングですが、ロールバインディングはプロジェクトレベルで存在します。ロールバインディングは、プロジェクトレベルで存在します。クラスターの **view (表示)** ロールは、ユーザーがプロジェクトを表示できるようローカルのロールバインディングを使用してユーザーにバインドする必要があります。ローカルロールは、クラスターのロールが特定の状況に必要なパーミッションのセットを提供しない場合にのみ作成する必要があります。

この 2 つのレベルからなる階層により、ローカルロールで個別プロジェクト内のカスタマイズが可能になる一方で、クラスターロールによる複数プロジェクト間での再利用が可能になります。

評価時に、クラスターロールのバインディングおよびローカルロールのバインディングが使用されます。以下に例を示します。

1. クラスター全体の allow ルールがチェックされます。
2. ローカルにバインドされた allow ルールがチェックされます。
3. デフォルトで拒否します。

5.1.1. デフォルトのクラスターロール

Red Hat OpenShift Service on AWS にはデフォルトのクラスターロールのセットがあります。このロールを、クラスター全体で、またはローカルにユーザーおよびグループにバインドできます。

重要

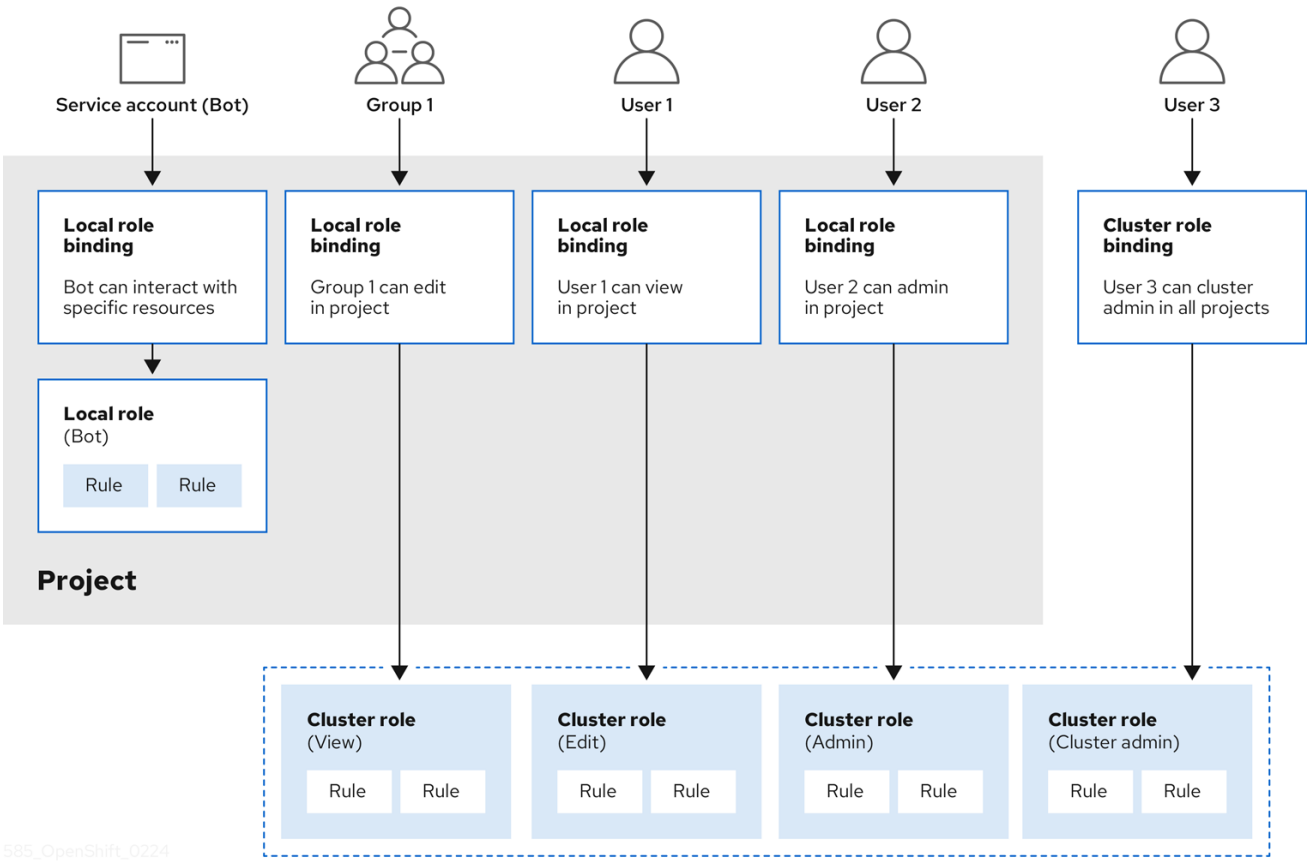
デフォルトのクラスターロールを手動で変更することは推奨されません。このようなシステムロールへの変更は、クラスターが正常に機能しなくなることがあります。

デフォルトのクラスターロール	説明
admin	プロジェクトマネージャー。ローカルバインディングで使用される場合、**admin** には、プロジェクト内のすべてのリソースを表示し、クォータ以外のリソース内のすべてのリソースを変更する権限があります。
basic-user	プロジェクトおよびユーザーについての基本的な情報を取得できるユーザーです。
cluster-admin	すべてのプロジェクトですべてのアクションを実行できるスーパーユーザーです。ローカルバインディングでユーザーにバインドされる場合、クォータに対する完全な制御およびプロジェクト内のすべてのリソースに対するすべてのアクションを実行できます。
cluster-status	基本的なクラスターのステータス情報を取得できるユーザーです。
cluster-reader	ほとんどのオブジェクトを取得または表示できるが、変更できないユーザー。
edit	プロジェクトのほとんどのプロジェクトを変更できるが、ロールまたはバインディングを表示したり、変更したりする機能を持たないユーザーです。
self-provisioner	独自のプロジェクトを作成できるユーザーです。
view	変更できないものの、プロジェクトでほとんどのオブジェクトを確認できるユーザーです。それらはロールまたはバインディングを表示したり、変更したりできません。

ローカルバインディングとクラスターバインディングについての違いに留意してください。ローカルのロールバインディングを使用して **cluster-admin** ロールをユーザーにバインドする場合、このユーザーがクラスター管理者の特権を持っているように表示されますが、実際にはそうではありません。一方、特定プロジェクトにバインドされる cluster-admin クラスターロールはそのプロジェクトのスーパー管理者のような機能があり、クラスターロール admin のパーミッションを付与するほか、レート制限を編集する機能などのいくつかの追加パーミッションを付与します。一方、**cluster-admin** をプロジェクト

のユーザーにバインドすると、そのプロジェクトにのみ有効なスーパー管理者の権限がそのユーザーに付与されます。そのユーザーはクラスターロール **admin** のパーミッションを有するほか、レート制限を編集する機能などの、そのプロジェクトについてのいくつかの追加パーミッションを持ちます。このバインディングは、クラスター管理者にバインドされるクラスターのロールバインディングをリスト表示しない Web コンソール UI を使うと分かりにくくなります。ただし、これは、**cluster-admin** をローカルにバインドするために使用するローカルのロールバインディングをリスト表示します。

クラスターロール、ローカルロール、クラスターロールのバインディング、ローカルロールのバインディング、ユーザー、グループおよびサービスアカウントの関係は以下に説明されています。



警告

get pods/exec、**get pods/***、および **get *** ルールは、ロールに適用されると実行権限を付与します。最小権限の原則を適用し、ユーザーおよびエージェントに必要な最小限の RBAC 権限のみを割り当てます。詳細は、RBAC ルールによる実行権限の許可 を参照してください。

5.1.2. 認可の評価

Red Hat OpenShift Service on AWS は、以下を使用して認可を評価します。

アイデンティティー

ユーザーが属するユーザー名とグループのリスト。

アクション

実行する動作。ほとんどの場合、これは以下で設定されます。

- **プロジェクト**: アクセスするプロジェクト。プロジェクトは追加のアノテーションを含む Kubernetes namespace であり、これにより、ユーザーのコミュニティーは、他のコミュニティーと分離された状態で独自のコンテンツを編成し、管理できます。
- **動詞**: **get**、**list**、**create**、**update**、**delete**、**deletecollection**、または **watch** などのアクション自体。
- **リソース名**: アクセスする API エンドポイント。

バインディング

バインディングの詳細なリスト、ロールを持つユーザーまたはグループ間の関連付け。

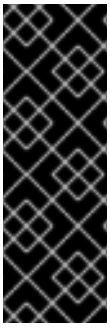
Red Hat OpenShift Service on AWS は以下の手順を使って認可を評価します。

1. アイデンティティーおよびプロジェクトでスコープ設定されたアクションは、ユーザーおよびそれらのグループに適用されるすべてのバインディングを検索します。
2. バインディングは、適用されるすべてのロールを見つけるために使用されます。
3. ロールは、適用されるすべてのルールを見つけるために使用されます。
4. 一致を見つけるために、アクションが各ルールに対してチェックされます。
5. 一致するルールが見つからない場合、アクションはデフォルトで拒否されます。

ヒント

ユーザーおよびグループは一度に複数のロールに関連付けたり、バインドしたりできることに留意してください。

プロジェクト管理者は CLI を使用してローカルロールとローカルバインディングを表示できます。これには、それぞれのロールが関連付けられる動詞およびリソースのマトリクスが含まれます。



重要

プロジェクト管理者にバインドされるクラスターロールは、ローカルバインディングによってプロジェクト内で制限されます。これは、**cluster-admin** または **system:admin** に付与されるクラスターロールのようにクラスター全体でバインドされる訳ではありません。

クラスターロールは、クラスターレベルで定義されるロールですが、クラスターレベルまたはプロジェクトレベルのいずれかでバインドできます。

5.1.2.1. クラスターロールの集計

デフォルトの admin、edit、view、cluster-reader クラスターロールでは、クラスターロールの集約 がサポートされており、各ロールは新規ルール作成時に動的に更新されます。この機能は、カスタムリソースを作成して Kubernetes API を拡張する場合にのみ適用できます。

5.2. プロジェクトおよび NAMESPACE

Kubernetes **namespace** は、クラスターでスコープ設定するメカニズムを提供します。namespace の詳細は、Kubernetes ドキュメント を参照してください。

Namespace は以下の一意のスコープを提供します。

- 基本的な命名の衝突を避けるための名前付きリソース。
- 信頼されるユーザーに委任された管理権限。
- コミュニティーリソースの消費を制限する機能。

システム内の大半のオブジェクトのスコープは namespace で設定されますが、一部はノードやユーザーを含め、除外され、namaspace が設定されません。

プロジェクト は追加のアノテーションを持つ Kubernetes namespace であり、通常ユーザーのリソースへのアクセスが管理される中心的な手段です。プロジェクトはユーザーのコミュニティーが他のコミュニティーとは切り離してコンテンツを編成し、管理することを許可します。ユーザーには、管理者によってプロジェクトへのアクセスが付与される必要があり、許可される場合はプロジェクトを作成でき、それらの独自のプロジェクトへのアクセスが自動的に付与されます。

プロジェクトには、別個の **name**、**displayName**、および **description** を設定できます。

- 必須の **name** はプロジェクトの一意の識別子であり、CLI ツールまたは API を使用する場合に最も表示されます。名前の最大長さは 63 文字です。
- オプションの **displayName** は、Web コンソールでのプロジェクトの表示方法を示します (デフォルトは **name** に設定される)。
- オプションの **description** には、プロジェクトのさらに詳細な記述を使用でき、これも Web コンソールで表示できます。

各プロジェクトは、以下の独自のセットのスコープを設定します。

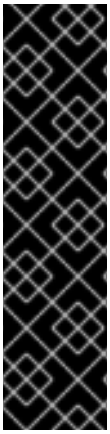
オブジェクト	説明
Objects	Pod、サービス、レプリケーションコントローラーなど。
Policies	ユーザーがオブジェクトに対してアクションを実行できるか、できないかについてのルール。
Constraints	制限を設定できるそれぞれの種類のオブジェクトのクォータ。
Service accounts	サービスアカウントは、プロジェクトのオブジェクトへの指定されたアクセスで自動的に機能します。

dedicated-admin ロールを持つ管理者は、プロジェクトを作成し、そのプロジェクトの管理者権限をユーザーコミュニティーのメンバーに委譲できます。また、**dedicated-admin** ロールを持つ管理者は、開発者が独自のプロジェクトを作成することを許可できます。

開発者および管理者は、CLI または Web コンソールを使用してプロジェクトとの対話を実行できます。

5.3. デフォルトプロジェクト

Red Hat OpenShift Service on AWS にはデフォルトのプロジェクトが多数含まれ、**openshift-** で始まるプロジェクトはユーザーにとって最も重要になります。これらのプロジェクトは、Pod として実行されるマスターコンポーネントおよび他のインフラストラクチャーコンポーネントをホストします。Critical Pod アノテーション を持つこれらの namespace で作成される Pod は Critical (重要) とみなされ、kubelet による受付が保証されます。これらの namespace のマスターコンポーネント用に作成された Pod には、すでに Critical のマークが付けられています。



重要

デフォルトプロジェクトでワークロードを実行したり、デフォルトプロジェクトへのアクセスを共有したりしないでください。デフォルトのプロジェクトは、コアクラスターコンポーネントを実行するために予約されています。

次のデフォルトプロジェクトは、高い特権があるとみなされます (**default**、**kube-public**、**kube-system**、**openshift**、**openshift-infra**、**openshift-node**、および **openshift.io/run-level** ラベルが **0** または **1** に設定されているその他のシステム作成プロジェクト)。Pod セキュリティーアドミッション、セキュリティーコンテキスト制約、クラスターリソースクォータ、イメージ参照解決などのアドミッションプラグインに依存する機能は、高い特権を持つプロジェクトでは機能しません。

5.4. クラスターロールおよびバインディングの表示

oc CLI で **oc describe** コマンドを使用して、クラスターロールおよびバインディングを表示できます。

前提条件

- **oc** CLI がインストールされている。
- クラスターロールおよびバインディングを表示するパーミッションを取得します。

手順

1. クラスターロールおよびそれらの関連付けられたルールセットを表示するには、以下を実行します。

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

出力例

```
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. 各種のロールにバインドされたユーザーおよびグループを示す、クラスターのロールバインディングの現在のセットを表示するには、以下を実行します。

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

出力例

```
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. ローカルのロールバインディングの表示

oc CLI で **oc describe** コマンドを使用して、ローカルロールおよびバインディングを表示できます。

前提条件

- **oc** CLI がインストールされている。
- ローカルロールおよびバインディングを表示するパーミッションを取得します。
  - ローカルにバインドされた **admin** のデフォルトのクラスターロールを持つユーザーは、そのプロジェクトのロールおよびバインディングを表示し、管理できます。

手順

1. 現在のプロジェクトの各種のロールにバインドされたユーザーおよびグループを示す、ローカルのロールバインディングの現在のセットを表示するには、以下を実行します。

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

2. 別のプロジェクトのローカルロールバインディングを表示するには、**-n** フラグをコマンドに追加します。

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

出力例

```
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. ロールのユーザーへの追加

oc adm 管理者 CLI を使用してロールおよびバインディングを管理できます。

ロールをユーザーまたはグループにバインドするか、追加することにより、そのロールによって付与されるアクセスがそのユーザーまたはグループに付与されます。**oc adm policy** コマンドを使用して、ロールのユーザーおよびグループへの追加、またはユーザーおよびグループからの削除を行うことができます。

デフォルトのクラスターロールのすべてを、プロジェクト内のローカルユーザーまたはグループにバインドできます。

手順

1. ロールを特定プロジェクトのユーザーに追加します。

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

たとえば、以下を実行して **admin** ロールを **joe** プロジェクトの **alice** ユーザーに追加できます。

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

ヒント

または、以下の YAML を適用してユーザーにロールを追加できます。

```
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. 出力でローカルロールバインディングを確認し、追加の内容を確認します。

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

たとえば、**joe** プロジェクトのローカルロールバインディングを表示するには、以下を実行します。

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

出力例

```
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** ユーザーが **admins RoleBinding** に追加されています。

5.7. ローカルロールの作成

プロジェクトのローカルロールを作成し、これをユーザーにバインドできます。

手順

1. プロジェクトのローカルロールを作成するには、以下のコマンドを実行します。

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

このコマンドで以下を指定します。

- **<name>**: ローカルのロール名です。
- **<verb>**: ロールに適用する動詞のコンマ区切りのリストです。
- **<resource>**: ロールが適用されるリソースです。
- **<project>** (プロジェクト名)

たとえば、ユーザーが **blue** プロジェクトで Pod を閲覧できるようにするローカルロールを作成するには、以下のコマンドを実行します。

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

2. 新規ロールをユーザーにバインドするには、以下のコマンドを実行します。

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

5.8. ローカルロールバインディングのコマンド

以下の操作を使用し、ローカルのロールバインディングでのユーザーまたはグループの関連付けられたロールを管理する際に、プロジェクトは **-n** フラグで指定できます。これが指定されていない場合には、現在のプロジェクトが使用されます。

ローカル RBAC 管理に以下のコマンドを使用できます。

表5.1 ローカルのロールバインディング操作

コマンド	説明
$ oc adm policy who-can <verb> <resource>	リソースに対してアクションを実行できるユーザーを示します。
$ oc adm policy add-role-to-user <role> <username>	指定されたロールを現在のプロジェクトの指定ユーザーにバインドします。
$ oc adm policy remove-role-from-user <role> <username>	現在のプロジェクトの指定ユーザーから指定されたロールを削除します。
$ oc adm policy remove-user <username>	現在のプロジェクトの指定ユーザーとそれらのロールのすべてを削除します。
$ oc adm policy add-role-to-group <role> <groupname>	指定されたロールを現在のプロジェクトの指定グループにバインドします。
$ oc adm policy remove-role-from-group <role> <groupname>	現在のプロジェクトの指定グループから指定されたロールを削除します。

コマンド	説明
$ oc adm policy remove-group <groupname>	現在のプロジェクトの指定グループとそれらのロールのすべてを削除します。

5.9. クラスターのロールバインディングコマンド

以下の操作を使用して、クラスターのロールバインディングも管理できます。クラスターのロールバインディングは namespace を使用していないリソースを使用するため、**-n** フラグはこれらの操作に使用されません。

表5.2 クラスターのロールバインディング操作

コマンド	説明
$ oc adm policy add-cluster-role-to-user <role> <username>	指定されたロールをクラスターのすべてのプロジェクトの指定ユーザーにバインドします。
$ oc adm policy remove-cluster-role-from-user <role> <username>	指定されたロールをクラスターのすべてのプロジェクトの指定ユーザーから削除します。
$ oc adm policy add-cluster-role-to-group <role> <groupname>	指定されたロールをクラスターのすべてのプロジェクトの指定グループにバインドします。
$ oc adm policy remove-cluster-role-from-group <role> <groupname>	指定されたロールをクラスターのすべてのプロジェクトの指定グループから削除します。

5.10. CLUSTER-ADMIN アクセス権限の付与

クラスターを作成したユーザーは、**cluster-admin** ユーザーロールをアカウントに追加して、最大管理者権限を割り当てます。これらの権限は、クラスターの作成時に自動的にユーザーアカウントに割り当てられることはありません。

さらに、クラスターを作成したユーザーのみが、他の **cluster-admin** または **dedicated-admin** ユーザーにクラスターアクセスを付与できます。**dedicated-admin** アクセスを持つユーザーの権限は少なくなります。ベストプラクティスとして、**cluster-admin** ユーザーの数をできるだけ少なく制限できます。

前提条件

- アイデンティティープロバイダー (IDP) をクラスターに追加している。
- 作成するユーザーの IDP ユーザー名がある。
- クラスターにログインしている。

手順

1. ユーザーに **cluster-admin** 権限を付与します。

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

2. ユーザーがクラスター管理者としてリスト表示されていることを確認します。

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

出力例

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

3. 以下のコマンドを実行して、ユーザーが **cluster-admin** アクセスを持つことを確認します。クラスター管理者はエラーを出さずにこのコマンドを実行できますが、専用の管理者は実行できません。

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

出力例

```
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. DEDICATED-ADMIN アクセスの取り消し

クラスターを作成したユーザーのみが、他の **cluster-admin** または **dedicated-admin** ユーザーにクラスターアクセスを付与できます。**dedicated-admin** アクセスを持つユーザーの権限は少なくなります。ベストプラクティスとして、**dedicated-admin** アクセスをほとんどの管理者に付与することができます。

前提条件

- アイデンティティープロバイダー (IDP) をクラスターに追加している。
- 作成するユーザーの IDP ユーザー名がある。
- クラスターにログインしている。

手順

1. 以下のコマンドを実行して、ユーザーを **dedicated-admin** にプロモートします。

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

2. 以下のコマンドを実行して、ユーザーに **dedicated-admin** アクセスがあることを確認します。

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

出力例

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

注記

Forbidden エラーは、**dedicated-admin** 権限を持たないユーザーがこのコマンドを実行する場合に表示されます。

第6章 サービスアカウントの概要および作成

6.1. サービスアカウントの概要

サービスアカウントは、コンポーネントが API に直接アクセスできるようにする Red Hat OpenShift Service on AWS アカウントです。サービスアカウントは各プロジェクト内に存在する API オブジェクトです。サービスアカウントは、通常ユーザーの認証情報を共有せずに API アクセスを制御する柔軟な方法を提供します。

Red Hat OpenShift Service on AWS CLI または Web コンソールを使用する場合、API トークンは API に対する認証を行います。コンポーネントをサービスアカウントに関連付け、通常ユーザーの認証情報を使用せずにそれらが API にアクセスできるようにします。

各サービスアカウントのユーザー名は、そのプロジェクトおよび名前から派生します。

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

すべてのサービスアカウントは 2 つのグループのメンバーでもあります。

グループ	説明
system:serviceaccounts	システムのすべてのサービスアカウントが含まれます。
system:serviceaccounts:<project>	指定されたプロジェクトのすべてのサービスアカウントが含まれます。

各サービスのアカウントには、2 つのシークレットが自動的に含まれます。

- API トークン
- OpenShift Container レジストリーの認証情報

生成される API トークンとレジストリーの認証情報は期限切れになることはありませんが、シークレットを削除することで取り消すことができます。シークレットが削除されると、新規のシークレットが自動生成され、これに置き換わります。

6.2. サービスアカウントの作成

サービスアカウントをプロジェクトで作成し、これをロールにバインドすることでパーミッションを付与できます。

手順

1. オプション: サービスアカウントを現在のプロジェクトで表示するには、以下を実行します。

```
$ oc get sa
```

出力例

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

```
deployer   2        2d
```

2. 新規サービスアカウントを現在のプロジェクトで作成するには、以下を実行します。

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

1 別のプロジェクトでサービスアカウントを作成するには、**-n <project_name>** を指定します。

出力例

```
serviceaccount "robot" created
```

ヒント

または、以下の YAML を適用してサービスアカウントを作成できます。

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

3. オプション: サービスアカウントのシークレットを表示します。

```
$ oc describe sa robot
```

出力例

```
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. ロールをサービスアカウントに付与する例

ロールをサービスアカウントに付与する方法は、ロールを通常ユーザーアカウントに付与する方法と同じです。

- 現在のプロジェクトのサービスアカウントを変更できます。たとえば、**view** ロールを **top-secret** プロジェクトの **robot** サービスアカウントに追加するには、以下を実行します。

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

ヒント

または、以下の YAML を適用してロールを追加できます。

```
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
```

- アクセスをプロジェクトの特定のサービスアカウントに付与することもできます。たとえば、サービスアカウントが属するプロジェクトから、**-z** フラグを使用し、**<service_account_name>** を指定します。

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

重要

プロジェクトの特定のサービスアカウントにアクセスを付与する必要がある場合には、**-z** フラグを使用します。このフラグを使用することにより、アクセスが指定されたサービスアカウントのみに付与することができます。

ヒント

または、以下の YAML を適用してロールを追加できます。

```
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>
```

- 別の namespace を変更するには、**-n** オプションを使用して、以下の例にあるように、適用先のプロジェクト namespace を指定します。
  - たとえば、すべてのプロジェクトのすべてのサービスアカウントが **my-project** プロジェクトのリソースを表示できるようにするには、以下を実行します。

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

ヒント

または、以下の YAML を適用してロールを追加できます。

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

- **managers** プロジェクトのすべてのサービスアカウントが **my-project** プロジェクトのリソースを編集できるようにするには、以下を実行します。

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

ヒント

または、以下の YAML を適用してロールを追加できます。

```
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 USING SERVICE ACCOUNTS IN APPLICATIONS

7.1. サービスアカウントの概要

サービスアカウントは、コンポーネントが API に直接アクセスできるようにする Red Hat OpenShift Service on AWS アカウントです。サービスアカウントは各プロジェクト内に存在する API オブジェクトです。サービスアカウントは、通常ユーザーの認証情報を共有せずに API アクセスを制御する柔軟な方法を提供します。

Red Hat OpenShift Service on AWS CLI または Web コンソールを使用する場合、API トークンは API に対する認証を行います。コンポーネントをサービスアカウントに関連付け、通常ユーザーの認証情報を使用せずにそれらが API にアクセスできるようにします。

各サービスアカウントのユーザー名は、そのプロジェクトおよび名前から派生します。

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

すべてのサービスアカウントは 2 つのグループのメンバーでもあります。

グループ	説明
system:serviceaccounts	システムのすべてのサービスアカウントが含まれます。
system:serviceaccounts:<project>	指定されたプロジェクトのすべてのサービスアカウントが含まれます。

各サービスのアカウントには、2 つのシークレットが自動的に含まれます。

- API トークン
- OpenShift Container レジストリーの認証情報

生成される API トークンとレジストリーの認証情報は期限切れになることはありませんが、シークレットを削除することで取り消すことができます。シークレットが削除されると、新規のシークレットが自動生成され、これに置き換わります。

7.2. デフォルトのサービスアカウント

Red Hat OpenShift Service on AWS クラスターには、クラスター管理用のデフォルトのサービスアカウントが含まれ、各プロジェクトのサービスアカウントは追加で生成されます。

7.2.1. デフォルトのクラスターサービスアカウント

一部のインフラストラクチャーコントローラーは、サービスアカウント認証情報を使用して実行されます。以下のサービスアカウントは、サーバーの起動時に Red Hat OpenShift Service on AWS インフラストラクチャープロジェクト (**openshift-infra**) に作成され、クラスター全体で以下のロールが付与されます。

サービスアカウント	説明
replication-controller	**system:replication-controller** ロールが割り当てられます。

サービスアカウント	説明
deployment-controller	**system:deployment-controller** ロールが割り当てられます。
build-controller	**system:build-controller** ロールが割り当てられます。さらに、**build-controller** サービスアカウントは、特権付きのビルド Pod を作成するために特権付きセキュリティーコンテキストに組み込まれます。

7.2.2. デフォルトのプロジェクトサービスアカウントおよびロール

3 つのサービスアカウントが各プロジェクトで自動的に作成されます。

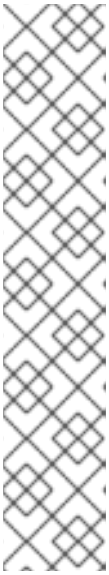
サービスアカウント	使用方法
builder	ビルド Pod で使用されます。これには **system:image-builder** ロールが付与されます。 このロールは、内部 Docker レジストリーを使用してイメージをプロジェクトのイメージストリームにプッシュすることを可能にします。
deployer	デプロイメント Pod で使用され、**system:deployer** ロールが付与されます。 このロールは、プロジェクトでレプリケーションコントローラーや Pod を表示したり、変更したりすることを可能にします。 **注記** **DeploymentConfig** クラスター機能が有効になっていない場合、**deployer** サービスアカウントは作成されません。
default	別のサービスアカウントが指定されていない限り、その他すべての Pod を実行するために使用されます。

プロジェクトのすべてのサービスアカウントには **system:image-puller** ロールが付与されます。このロールがあることで、内部コンテナーイメージレジストリーを使用してイメージをイメージストリームからプルできます。

7.2.3. 自動生成されたシークレット

デフォルトでは、Red Hat OpenShift Service on AWS はサービスアカウントごとに次のシークレットを作成します。

- dockercfg イメージプルシークレット
- サービスアカウントトークンシークレット



注記

Red Hat OpenShift Service on AWS 4.11 より前では、サービスアカウントの作成時に 2 つ目のサービスアカウントトークンシークレットが生成されていました。このサービスアカウントトークンシークレットは、Kubernetes API へのアクセスに使用されていました。

Red Hat OpenShift Service on AWS 4.11 以降、この 2 つ目のサービスアカウントトークンシークレットは作成されなくなりました。これは、**LegacyServiceAccountTokenNoAutoGeneration** アップストリーム Kubernetes フィーチャーゲートが有効になっており、Kubernetes API にアクセスするためのシークレットベースのサービスアカウントトークンの自動生成が停止されているためです。

4 にアップグレードした後も、既存のサービスアカウントトークンシークレットは削除されず、引き続き機能します。

このサービスアカウントトークンシークレットと Docker 設定イメージプルシークレットは、OpenShift イメージレジストリーをクラスターのユーザー認証および認可システムに統合するために必要です。

ただし、**ImageRegistry** 機能を有効にしていない場合、または Cluster Image Registry Operator の設定で統合された OpenShift イメージレジストリーを無効にしている場合、これらのシークレットはサービスアカウントごとに生成されません。

警告

自分で使用する場合は、これらの自動生成されたシークレットに依存しないようにしてください。これらは、今後の Red Hat OpenShift Service on AWS のリリースで削除される可能性があります。

バインドされたサービスアカウントトークンを取得するために、予測されたボリュームでワークロードが自動的に挿入されます。ワークロードに追加のサービスアカウントトークンが必要な場合は、ワークロードマニフェストに追加の予測ボリュームを追加します。バインドされたサービスアカウントトークンは、次の理由により、サービスアカウントトークンのシークレットよりも安全です。

- バインドされたサービスアカウントトークンには有効期間が制限されています。
- バインドされたサービスアカウントトークンには対象ユーザーが含まれます。
- バインドされたサービスアカウントトークンは Pod またはシークレットにバインドでき、バインドされたオブジェクトが削除されるとバインドされたトークンは無効になります。

詳細は、**ボリュームプロジェクションを使用したバインドされたサービスアカウントトークンの設定** を参照してください。

読み取り可能な API オブジェクト内の有効期限のないトークンのセキュリティー露出が許容される場合は、サービスアカウントトークンシークレットを手動で作成してトークンを取得することもできます。詳細は、**サービスアカウントトークンシークレットの作成** を参照してください。

関連情報

- バインドされたサービスアカウントトークンのリクエストについては、ボリュームプロジェクションを使用したバインドされたサービスアカウントトークンの設定 を参照してください。
- サービスアカウントトークンシークレットの作成については、サービスアカウントトークンシークレットの作成 を参照してください。

7.3. サービスアカウントの作成

サービスアカウントをプロジェクトで作成し、これをロールにバインドすることでパーミッションを付与できます。

手順

1. オプション: サービスアカウントを現在のプロジェクトで表示するには、以下を実行します。

```
$ oc get sa
```

出力例

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

2. 新規サービスアカウントを現在のプロジェクトで作成するには、以下を実行します。

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

1 別のプロジェクトでサービスアカウントを作成するには、**-n <project_name>** を指定します。

出力例

```
serviceaccount "robot" created
```

ヒント

または、以下の YAML を適用してサービスアカウントを作成できます。

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

3. オプション: サービスアカウントのシークレットを表示します。

```
$ oc describe sa robot
```

出力例

```
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 サービスアカウントの OAUTH クライアントとしての使用

8.1. OAUTH クライアントとしてのサービスアカウント

サービスアカウントは、OAuth クライアントの制限されたフォームで使用できます。サービスアカウントは一部の基本ユーザー情報へのアクセスを許可するスコープのサブセットと、サービスアカウント自体の namespace 内のロールベースの権限のみを要求できます。

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

サービスアカウントを OAuth クライアントとして使用する場合:

- **client_id** は **system:serviceaccount:<service_account_namespace>:<service_account_name>** です。
- **client_secret** には、サービスアカウントの API トークンのいずれかを指定できます。以下に例を示します。

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

- **WWW-Authenticate** チャレンジを取得するには、サービスアカウントの **serviceaccounts.openshift.io/oauth-want-challenges** アノテーションを **true** に設定します。
- **redirect_uri** は、サービスアカウントのアノテーションに一致する必要があります。

8.1.1. OAuth クライアントとしてのサービスアカウントの URI のリダイレクト

アノテーションキーには、以下のように接頭辞 **serviceaccounts.openshift.io/oauth-redirecturi.** または **serviceaccounts.openshift.io/oauth-redirectreference.** が含まれる必要があります。

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

最も単純なフォームでは、アノテーションは有効なリダイレクト URI を直接指定するために使用できます。以下に例を示します。

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

上記の例の **first** および **second** ポストフィックスは 2 つの有効なリダイレクト URI を分離するために使用されます。

さらに複雑な設定では、静的なリダイレクト URI のみでは不十分な場合があります。たとえば、ルートのすべての ingress が有効とみなされる必要があるかもしれません。この場合、**serviceaccounts.openshift.io/oauth-redirectreference.** 接頭辞を使用した動的なリダイレクト URI を使用できます。

以下に例を示します。

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

このアノテーションの値にはシリアライズされた JSON データが含まれるため、これを拡張フォーマットで表示するとより容易になります。

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

ここでは、**OAuthRedirectReference** により **jenkins** という名前のルートを参照できます。そのため、そのルートのすべての ingress は有効とみなされます。**OAuthRedirectReference** の詳細な仕様は以下のようになります。

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

1. **kind** は参照されているオブジェクトのタイプを参照します。現時点では、**route** のみがサポートされています。
2. **name** はオブジェクトの名前を参照します。このオブジェクトはサービスアカウントと同じ namespace にある必要があります。
3. **group** はオブジェクトのグループを参照します。ルートのグループは空の文字列であるため、これを空白のままにします。

アノテーションはどちらも、接頭辞も組み合わせて、参照オブジェクトで提供されるデータをオーバーライドできます。以下に例を示します。

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

first ポストフィックスはアノテーションを関連付けるために使用されます。**jenkins** ルートに **https://example.com** の Ingress がある場合に、**https://example.com/custompath** は有効とみなされますが、**https://example.com** は有効とみなされません。オーバーライドするデータを部分的に指定するためのフォーマットは以下のようになります。

タイプ	構文
スキーム	"https://"
ホスト名	"//website.com"
ポート	"//:8000"
パス	"examplepath"

注記

ホスト名のオーバーライドを指定すると、参照されるオブジェクトのホスト名データが置き換わりますが、これは望ましい動作ではありません。

上記の構文のいずれの組み合わせも、以下のフォーマットを使用して実行できます。

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

同じオブジェクトを複数回参照して、柔軟性を向上することができます。

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

jenkins という名前のルートに **https://example.com** の Ingress がある場合には、**https://example.com:8000** と **https://example.com/custompath** の両方が有効とみなされます。

必要な動作を得るために、静的で動的なアノテーションを同時に使用できます。

```
"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 サービスアカウントの AWS IAM ロールの引き受け

AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS では、OpenShift API サーバーを有効にして、Pod 内の AWS Identity and Access Management (IAM) ロールの引き受けに使用できる署名付きサービスアカウントトークンを投影できます。想定した IAM ロールに必要な AWS パーミッションがある場合は、Pod は、一時的な STS 認証を使用して、AWS 操作を実行し、AWS API に対して認証できます。

Pod ID Webhook を使用してサービスアカウントトークンを生成し、独自のワークロードに対する AWS Identity and Access Management (IAM) ロールを推測できます。想定された IAM ロールに必要な AWS アクセス許可がある場合、Pod は一時的な STS 認証情報を使用して AWS SDK 操作を実行できます。

9.1. サービスアカウントが SRE 所有のプロジェクトで AWS IAM ロールを引き受ける方法

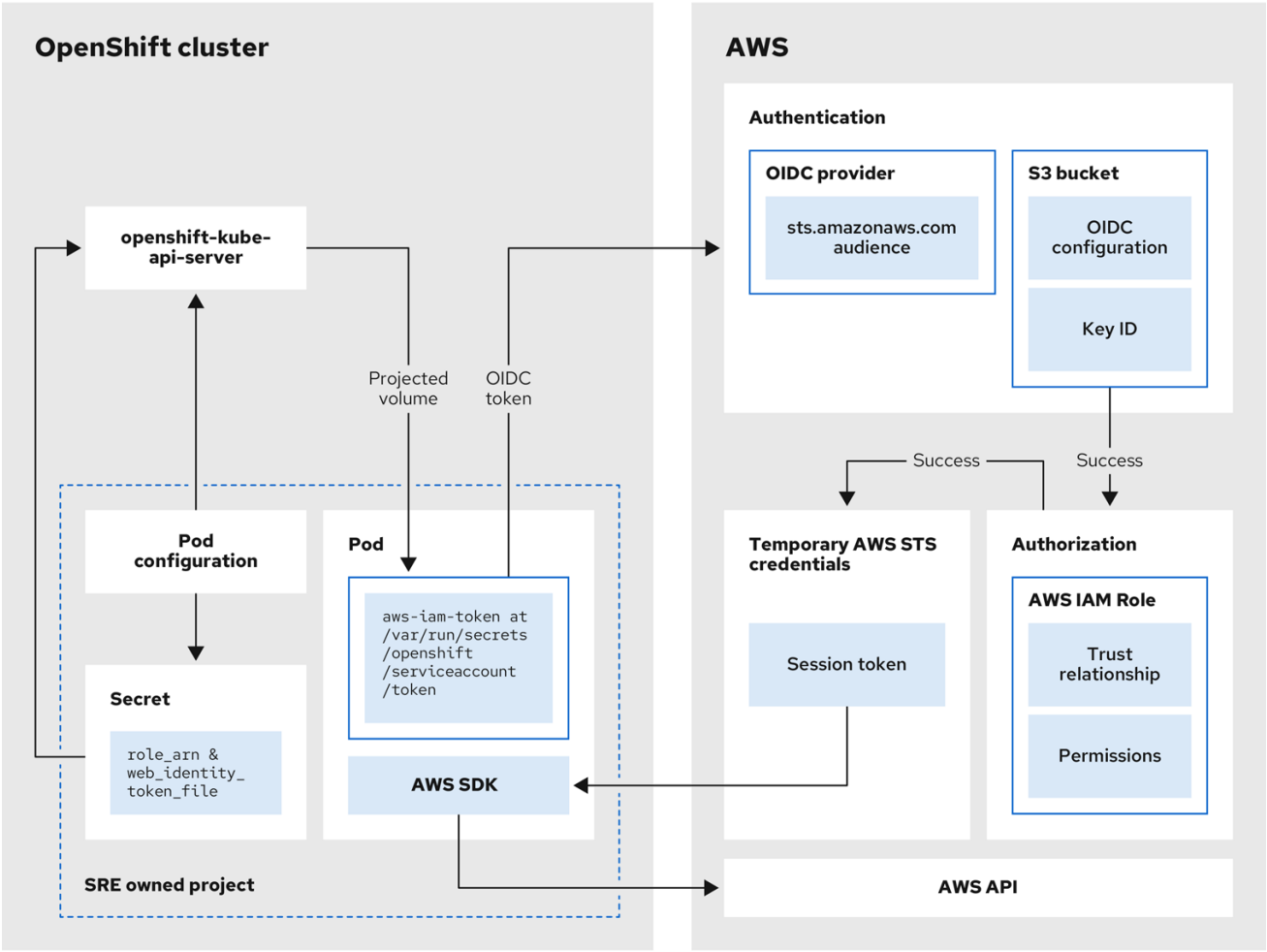
AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS クラスターをインストールすると、クラスター固有の Operator AWS Identity and Access Management (IAM) ロールが作成されます。これらの IAM ロールにより、Red Hat OpenShift Service on AWS クラスター Operator がコア OpenShift 機能を実行できるようになります。

クラスター Operator はサービスアカウントを使用して IAM ロールを引き受けます。サービスアカウントが IAM ロールを引き受けると、クラスター Operator の Pod で使用するサービスアカウントに一時的な STS 認証情報が提供されます。引き受けたロールに必要な AWS 権限がある場合、サービスアカウントは Pod で AWS SDK 操作を実行できます。

SRE 所有プロジェクトで AWS IAM ロールを引き受けるワークフロー

次の図は、SRE 所有プロジェクトで AWS IAM ロールを引き受けるためのワークフローを示しています。

図9.1 SRE 所有プロジェクトで AWS IAM ロールを引き受けるワークフロー



ワークフローには次の段階があります。

1. クラスター Operator が実行する各プロジェクト内で、Operator のデプロイメント仕様には、投影されたサービスアカウントトークンのボリュームマウントと、Pod の AWS 認証情報設定が含まれるシークレットがあります。トークンは、オーディエンスおよび時間の制限があります。Red Hat OpenShift Service on AWS は 1 時間ごとに新しいトークンを生成し、AWS SDK は AWS 認証情報の設定を含むマウントされたシークレットを読み取ります。この設定には、マウントされたトークンと AWS IAM ロール ARN へのパスが含まれています。シークレットの認証情報設定には次のものが含まれます。

    - AWS SDK オペレーションの実行に必要なパーミッションを持つ IAM ロールの ARN を含む **$AWS_ARN_ROLE** 変数。

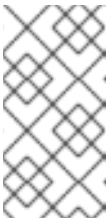
    - サービスアカウントの OpenID Connect (OIDC) トークンへの Pod 内のフルパスを含む **$AWS_WEB_IDENTITY_TOKEN_FILE** 変数。完全パスは **/var/run/secrets/openshift/serviceaccount/token** です。

2. クラスター Operator が AWS サービス(EC2 など)にアクセスするために AWS IAM ロールを引き受ける必要がある場合、Operator で実行される AWS SDK クライアントコードは **AssumeRoleWithWebIdentity** API を呼び出します。

3. OIDC トークンは、Pod から OIDC プロバイダーに渡されます。次の要件が満たされている場合は、プロバイダーがサービスアカウント ID を認証します。

    - ID 署名は有効であり、秘密鍵によって署名されています。

- **sts.amazonaws.com** オーディエンスは OIDC トークンにリストされており、OIDC プロバイダーで設定されたオーディエンスと一致します。



注記

STS クラスターを使用する Red Hat OpenShift Service on AWS では、インストール中に OIDC プロバイダーが作成され、デフォルトでサービスアカウント発行者として設定されます。**sts.amazonaws.com** オーディエンスは、デフォルトで OIDC プロバイダーに設定されています。

- OIDC トークンの有効期限が切れていません。
- トークン内の発行者の値には、OIDC プロバイダーの URL が含まれています。

4. プロジェクトとサービスアカウントが、引き受ける IAM ロールの信頼ポリシーのスコープ内にある場合は、認可が成功します。
5. 認証と認可が成功すると、AWS アクセストークン、秘密鍵、セッショントークンの形式で一時的な AWS STS 認証情報が Pod に渡され、サービスアカウントで使用されます。認証情報を使用することで、IAM ロールで有効になっている AWS アクセス許可がサービスアカウントに一時的に付与されます。
6. クラスター Operator が実行されると、Pod で AWS SDK を使用している Operator は、投影されたサービスアカウントへのパスが含まれるシークレットと AWS IAM ロール ARN を OIDC プロバイダーに対して認証するためのシークレットを消費します。OIDC プロバイダーは、AWS API に対する認証に使用できるように、一時的な STS 認証情報を返します。

9.2. サービスアカウントがユーザー定義プロジェクトで AWS IAM ロールを引き受ける方法

AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS クラスターをインストールすると、Pod ID Webhook リソースがデフォルトで含まれます。

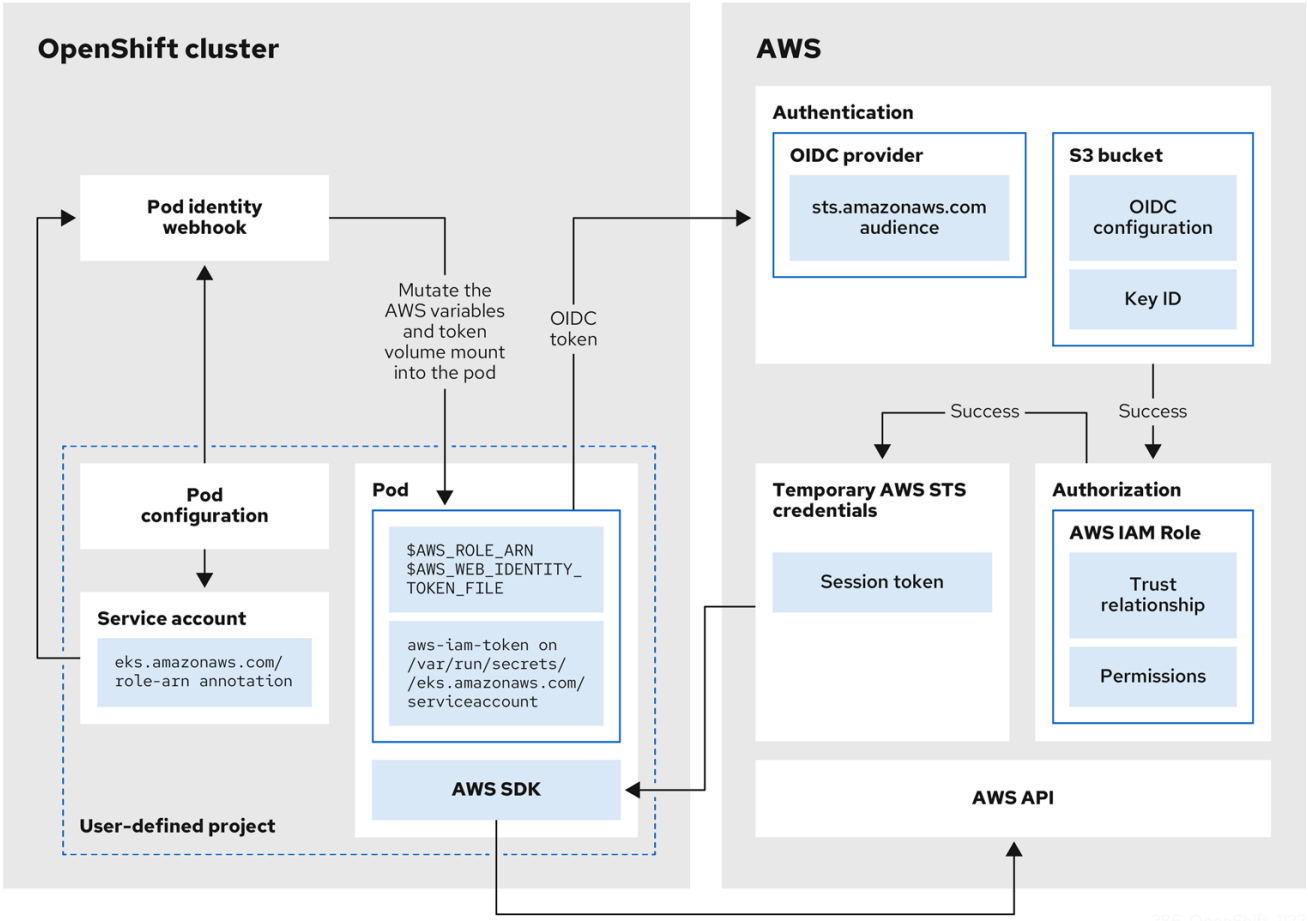
Pod ID Webhook を使用して、ユーザー定義プロジェクトのサービスアカウントが、同じプロジェクトの Pod で AWS Identity and Access Management (IAM) ロールを引き受けることができます。IAM ロールを引き受けると、Pod 内のサービスアカウントが使用する一時的な STS 認証情報が提供されます。引き受けたロールに必要な AWS 権限がある場合、サービスアカウントは Pod で AWS SDK 操作を実行できます。

Pod の Pod ID Webhook を有効にするには、プロジェクトで **eks.amazonaws.com/role-arn** アノテーションを使用してサービスアカウントを作成する必要があります。アノテーションは、サービスアカウントが引き受ける AWS IAM ロールの Amazon Resource Name (ARN) を参照する必要があります。また、**Pod** 仕様でサービスアカウントを参照し、サービスアカウントと同じプロジェクトに Pod をデプロイする必要があります。

ユーザー定義プロジェクトでの Pod ID Webhook ワークフロー

次の図は、ユーザー定義プロジェクトでの Pod ID Webhook ワークフローを示しています。

図9.2 ユーザー定義プロジェクトでの Pod ID Webhook ワークフロー



ワークフローには次の段階があります。

1. ユーザー定義のプロジェクト内で、ユーザーは **eks.amazonaws.com/role-arn** アノテーションを含むサービスアカウントを作成します。アノテーションは、サービスアカウントが引き受ける AWS IAM ロールの ARN を指します。

2. アノテーション付きのサービスアカウントを参照する設定を使用して Pod が同じプロジェクトにデプロイされると、Pod ID Webhook により Pod が変更になります。ミューテーションは、**Pod** または **Deployment** リソース設定で指定する必要なく、次のコンポーネントを Pod に挿入します。

   - AWS SDK オペレーションを実行するために必要なアクセス権を持つ IAM ロールの ARN を含む **$AWS_ARN_ROLE** 環境変数。

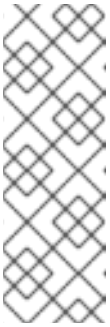
   - サービスアカウントの OpenID Connect (OIDC) トークンへの Pod 内のフルパスを含む **$AWS_WEB_IDENTITY_TOKEN_FILE** 環境変数。フルパスは **/var/run/secrets/eks.amazonaws.com/serviceaccount/token** です。

   - マウントポイント **/var/run/secrets/eks.amazonaws.com/serviceaccount** にマウントされた **aws-iam-token** ボリューム。**token** という名前の OIDC トークンファイルがボリュームに含まれています。

3. OIDC トークンは、Pod から OIDC プロバイダーに渡されます。次の要件が満たされている場合は、プロバイダーがサービスアカウント ID を認証します。

   - ID 署名は有効であり、秘密鍵によって署名されています。

- **sts.amazonaws.com** オーディエンスは OIDC トークンにリストされており、OIDC プロバイダーで設定されたオーディエンスと一致します。



注記

Pod ID Webhook は、デフォルトで **sts.amazonaws.com** オーディエンスを OIDC トークンに適用します。

STS クラスターを使用する Red Hat OpenShift Service on AWS では、インストール中に OIDC プロバイダーが作成され、デフォルトでサービスアカウント発行者として設定されます。**sts.amazonaws.com** オーディエンスは、デフォルトで OIDC プロバイダーに設定されています。

- OIDC トークンの有効期限が切れていません。
- トークンの発行者の値には、OIDC プロバイダーの URL が含まれています。

4. プロジェクトとサービスアカウントが、引き受ける IAM ロールの信頼ポリシーのスコープ内にある場合は、認可が成功します。
5. 認証と認可が成功すると、セッショントークンの形式の一時的な AWS STS 認証情報が Pod に渡され、サービスアカウントで使用できるようになります。認証情報を使用することで、IAM ロールで有効になっている AWS アクセス許可がサービスアカウントに一時的に付与されます。
6. Pod で AWS SDK オペレーションを実行すると、サービスアカウントは一時的な STS 認証情報を AWS API に提供して、その ID を確認します。

9.3. 独自の POD で AWS IAM ロールを引き受ける

このセクションの手順に従って、ユーザー定義のプロジェクトにデプロイされた Pod でサービスアカウントが AWS Identity and Access Management (IAM) ロールを引き受けられるようにします。

AWS IAM ロール、サービスアカウント、AWS SDK を含むコンテナーイメージ、イメージを使用してデプロイされた Pod など、必要なリソースを作成できます。この例では、AWS Boto3 SDK for Python が使用されています。また、Pod ID Webhook が AWS 環境変数、ボリュームマウント、およびトークンボリュームを Pod に変更することを確認することもできます。さらに、サービスアカウントが Pod で AWS IAM ロールを引き受け、AWS SDK オペレーションを正常に実行できることを確認できます。

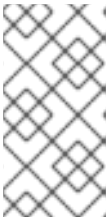
9.3.1. サービスアカウントの AWS IAM ロールの設定

Red Hat OpenShift Service on AWS クラスターのサービスアカウントが引き受ける AWS Identity and Access Management (IAM) ロールを作成します。サービスアカウントが Pod で AWS SDK オペレーションを実行するために必要なアクセス許可をアタッチします。

前提条件

- AWS アカウントに IAM ロールをインストールして設定するために必要なアクセス許可がある。
- AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS クラスターにアクセスできる。管理者レベルのユーザー権限は必要ありません。

- STS クラスターを使用する Red Hat OpenShift Service on AWS でサービスアカウント発行者として設定されている OpenID Connect (OIDC) プロバイダーの Amazon Resource Name (ARN) がある。



注記

STS クラスターを使用する Red Hat OpenShift Service on AWS では、インストール中に OIDC プロバイダーが作成され、デフォルトでサービスアカウント発行者として設定されます。OIDC プロバイダーの ARN がわからない場合は、クラスター管理者に問い合わせてください。

- AWS CLI (**aws**) をインストールしている。

手順

1. 次の JSON 設定を使用して、**trust-policy.json** という名前のファイルを作成します。

```
{
   "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. **<oidc_provider_arn>** を OIDC プロバイダーの ARN に置き換えます (例: **arn:aws:iam::<aws_account_id>:oidc-provider/rh-oidc.s3.us-east-1.amazonaws.com/1v3r0n44npxu4g58so46aeohduomfres**)。

2. 指定されたプロジェクトとサービスアカウントにロールを制限します。**<oidc_provider_name>** を OIDC プロバイダーの名前に置き換えます (例: **rh-oidc.s3.us-east-1.amazonaws.com/1v3r0n44npxu4g58so46aeohduomfres**)。**<project_name>:<service_account_name>** を実際のプロジェクト名とサービスアカウント名に置き換えます (例: **my-project:test-service-account**)。

注記

または、**"<oidc_provider_name>:sub": "system:serviceaccount:<project_name>:*"** を使用して、指定したプロジェクト内の任意のサービスアカウントにロールを制限できます。* ワイルドカードを指定する場合は、前の行で **StringEquals** を **StringLike** に置き換える必要があります。

2. **trust-policy.json** ファイルで定義されている信頼ポリシーを使用する AWS IAM ロールを作成します。

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

1. **<aws_iam_role_name>** を **pod-identity-test-role** などの IAM ロールの名前に置き換えます。
2. 前の手順で作成した **trust-policy.json** ファイルを参照します。

出力例:

```
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>
```

出力でロールの ARN を保持します。ロール ARN の形式は **arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name>** です。

3. サービスアカウントが Pod で AWS SDK 操作を実行するときに必要なマネージド AWS アクセス許可をアタッチします。

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

1. この例のポリシーは、IAM ロールに読み取り専用アクセス許可を追加します。
2. **<aws_iam_role_name>** を、前のステップで作成した IAM ロールの名前に置き換えます。

4. オプション: カスタム属性または権限境界をロールに追加します。詳細は、AWS ドキュメントの AWS サービスにアクセス許可を委任するロールの作成 を参照してください。

9.3.2. プロジェクトでサービスアカウントを作成する

ユーザー定義プロジェクトにサービスアカウントを追加します。サービスアカウントに引き受けさせる AWS Identity and Access Management (IAM) ロールの Amazon Resource Name (ARN) を参照する **eks.amazonaws.com/role-arn** アノテーションをサービスアカウント設定に含めます。

前提条件

- サービスアカウントの AWS IAM ロールを作成している。詳細は、**サービスアカウントの AWS IAM ロールの設定** を参照してください。
- AWS Security Token Service (STS) クラスターを使用して Red Hat OpenShift Service on AWS にアクセスできます。管理者レベルのユーザー権限は必要ありません。

- OpenShift CLI (**oc**) がインストールされている。

手順

1. Red Hat OpenShift Service on AWS クラスターで、プロジェクトを作成します。

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

1 **<project-name>** は、プロジェクト名に置き換えます。この名前は、AWS IAM ロールの設定で指定したプロジェクト名と一致する必要があります。

注記

プロジェクトが作成されると、自動的にプロジェクトに切り替わります。

2. 次のサービスアカウント設定で、**test-service-account.yaml** という名前のファイルを作成します。

```
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 **<service_account_name>** をサービスアカウントの名前に置き換えます。この名前は、AWS IAM ロールの設定で指定したサービスアカウント名と一致させる必要があります。

2 **<project-name>** は、プロジェクト名に置き換えます。この名前は、AWS IAM ロールの設定で指定したプロジェクト名と一致する必要があります。

3 サービスアカウントが Pod 内で使用するために想定する AWS IAM ロールの ARN を指定します。**<aws_iam_role_arn>** を、サービスアカウント用に作成した AWS IAM ロールの ARN に置き換えます。ロール ARN の形式は **arn:aws:iam::<aws_account_id>:role/<aws_iam_role_name>** です。

3. プロジェクトにサービスアカウントを作成します。

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

出力例:

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

4. サービスアカウントの詳細を確認します。

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

1 **<service_account_name>** をサービスアカウントの名前に置き換えます。

出力例:

```
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. サービスアカウントの名前を指定します。
2. サービスアカウントを含むプロジェクトを指定します。
3. サービスアカウントが引き受ける AWS IAM ロールの ARN のアノテーションを一覧表示します。

9.3.3. サンプル AWS SDK コンテナーイメージの作成

この手順のステップでは、AWS SDK を含むコンテナーイメージを作成する方法の例を示します。

例の手順では、Podman を使用してコンテナーイメージを作成し、Quay.io を使用してイメージをホストします。Quay.io の詳細は、Quay.io の使用を開始する を参照してください。コンテナーイメージを使用して、AWS SDK オペレーションを実行できる Pod をデプロイできます。

注記

この手順例では、AWS Boto3 SDK for Python がコンテナーイメージにインストールされます。AWS Boto3 SDK のインストールと使用の詳細は、AWS Boto3 ドキュメント を参照してください。その他の AWS SDK の詳細は、AWS ドキュメントの AWS SDKs and Tools Reference Guide を参照してください。

前提条件

- インストールホストに Podman をインストールしている。
- Quay.io ユーザーアカウントを持っている。

手順

1. 次の設定を **Containerfile** という名前のファイルに追加します。

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

1. Red Hat Universal Base Image バージョン 9 を指定します。
2. **pip** パッケージ管理システムを使用して AWS Boto3 SDK をインストールします。この例では、AWS Boto3 SDK バージョン 1.15.0 以降がインストールされています。

2. ファイルを含むディレクトリーから、**awsboto3sdk** という名前のコンテナーイメージをビルドします。

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

3. Quay.io にログインします。

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

4. Quay.io へのアップロードの準備として、イメージにタグを付けます。

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

1 **<quay_username>** を Quay.io ユーザー名に置き換えます。

5. タグ付けされたコンテナーイメージを Quay.io にプッシュします。

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

1 **<quay_username>** を Quay.io ユーザー名に置き換えます。

6. イメージを含む Quay.io リポジトリーを公開します。これによりイメージが公開され、Red Hat OpenShift Service on AWS クラスターに Pod をデプロイするために使用できるようになります。

   a. https://quay.io/ で、イメージを含むリポジトリーの **Repository Settings** ページに移動します。

   b. **Make Public** をクリックして、リポジトリーを公開します。

9.3.4. AWS SDK を含む Pod のデプロイ

AWS SDK を含むコンテナーイメージからユーザー定義プロジェクトに Pod をデプロイします。Pod 設定で、**eks.amazonaws.com/role-arn** アノテーションを含むサービスアカウントを指定します。

Pod のサービスアカウント参照が配置されると、Pod ID Webhook が AWS 環境変数、ボリュームマウント、およびトークンボリュームを Pod に挿入します。Pod のミューテーションにより、サービスアカウントは Pod で AWS IAM ロールを自動的に引き受けることができます。

前提条件

- サービスアカウントの AWS Identity and Access Management (IAM) ロールを作成している。詳細は、**サービスアカウントの AWS IAM ロールの設定**を参照してください。
- AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS クラスターにアクセスできる。管理者レベルのユーザー権限は必要ありません。
- OpenShift CLI (**oc**) がインストールされている。
- サービスアカウントが引き受ける IAM ロールの Amazon Resource Name (ARN) を参照する **eks.amazonaws.com/role-arn** アノテーションを含むサービスアカウントをプロジェクトに作成している。

- AWS SDK を含むコンテナーイメージがあり、そのイメージをクラスターで使用できる。詳細な手順は、**サンプルの AWS SDK コンテナーイメージの作成** を参照してください。

注記

この手順例では、AWS Boto3 SDK for Python が使用されています。AWS Boto3 SDK のインストールと使用の詳細は、AWS Boto3 ドキュメント を参照してください。その他の AWS SDK の詳細は、AWS ドキュメントの AWS SDKs and Tools Reference Guide を参照してください。

手順

1. 次の Pod 設定で **awsboto3sdk-pod.yaml** という名前のファイルを作成します。

```
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. **<project-name>** は、プロジェクト名に置き換えます。この名前は、AWS IAM ロールの設定で指定したプロジェクト名と一致する必要があります。
2. Pod の名前を指定します。
3. **<service_account_name>** を、AWS IAM ロールを引き受けるように設定されたサービスアカウントの名前に置き換えます。この名前は、AWS IAM ロールの設定で指定したサービスアカウント名と一致させる必要があります。
4. **awsboto3sdk** コンテナーイメージの場所を指定します。**<quay_username>** を Quay.io ユーザー名に置き換えます。
5. この Pod 設定の例では、この行は Pod を 100000 秒間実行し続け、Pod で直接検証テストを有効にします。詳細な検証手順は、**Pod で想定される IAM ロールの検証** を参照してください。

2. **awsboto3sdk** Pod をデプロイします。

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

出力例:

```
pod/awsboto3sdk created
```

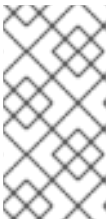
9.3.5. Pod で想定される IAM ロールを確認する

プロジェクトに **awsboto3sdk** Pod をデプロイした後、Pod ID Webhook が Pod を変更したことを確認します。必要な AWS 環境変数、ボリュームマウント、および OIDC トークンボリュームが Pod 内に存在することを確認します。

Pod で AWS SDK オペレーションを実行するときに、サービスアカウントが AWS アカウントの AWS Identity and Access Management (IAM) ロールを引き受けていることを確認することもできます。

前提条件

- サービスアカウントの AWS IAM ロールを作成している。詳細は、**サービスアカウントの AWS IAM ロールの設定** を参照してください。
- AWS Security Token Service (STS) を使用する Red Hat OpenShift Service on AWS クラスターにアクセスできる。管理者レベルのユーザー権限は必要ありません。
- OpenShift CLI (**oc**) がインストールされている。
- サービスアカウントが引き受ける IAM ロールの Amazon Resource Name (ARN) を参照する **eks.amazonaws.com/role-arn** アノテーションを含むサービスアカウントをプロジェクトに作成している。
- AWS SDK を含むユーザー定義プロジェクトに Pod をデプロイしている。Pod は、Pod ID Webhook を使用するサービスアカウントを参照して、AWS SDK オペレーションの実行に必要な AWS IAM ロールを引き受けます。詳細な手順は、**AWS SDK を含む Pod のデプロイ** を参照してください。



注記

この手順例では、AWS Boto3 SDK for Python を含む Pod が使用されます。AWS Boto3 SDK のインストールと使用の詳細は、AWS Boto3 ドキュメント を参照してください。その他の AWS SDK の詳細は、AWS ドキュメントの AWS SDKs and Tools Reference Guide を参照してください。

手順

1. AWS 環境変数、ボリュームマウント、および OIDC トークンボリュームが、デプロイされた **awsboto3sdk** Pod の説明にリストされていることを確認します。

```
$ oc describe pod awsboto3sdk
```

出力例:

```
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. Pod ID Webhook によって Pod に挿入された **AWS_ROLE_ARN** 環境変数を一覧表示します。この変数には、サービスアカウントが引き受ける AWS IAM ロールの ARN が含まれます。
2. Pod ID Webhook によって Pod に挿入された **AWS_WEB_IDENTITY_TOKEN_FILE** 環境変数を一覧表示します。この変数には、サービスアカウントの ID を確認するために使用される OIDC トークンの完全なパスが含まれています。
3. Pod ID Webhook によって Pod に挿入されたボリュームマウントを一覧表示します。
4. **/var/run/secrets/eks.amazonaws.com/serviceaccount** マウントポイントにマウントされている **aws-iam-token** ボリュームを一覧表示します。ボリュームには、AWS IAM ロールを引き受けるためにサービスアカウントを認証するために使用される OIDC トークンが含まれています。

2. **awsboto3sdk** Pod でインタラクティブターミナルを起動します。

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

3. Pod のインタラクティブターミナルで、Pod ID Webhook により **$AWS_ROLE_ARN** 環境変数が Pod に変更されたことを確認します。

```
$ echo $AWS_ROLE_ARN
```

出力例:

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

1. 出力では、AWS SDK オペレーションを実行するために必要なアクセス許可を持つ AWS IAM ロールの ARN を指定する必要があります。

4. Pod のインタラクティブターミナルで、Pod ID Webhook によって **$AWS_WEB_IDENTITY_TOKEN_FILE** 環境変数が Pod に変更されたことを確認します。

```
$ echo $AWS_WEB_IDENTITY_TOKEN_FILE
```

出力例:

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

1 出力では、Pod 内のサービスアカウントの OIDC トークンへのフルパスを指定する必要があります。

5. Pod のインタラクティブターミナルで、OIDC トークンファイルを含む **aws-iam-token** ボリュームマウントが Pod ID Webhook によってマウントされたことを確認します。

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

出力例:

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

6. Pod のインタラクティブターミナルで、**token** という名前の OIDC トークンファイルが **/var/run/secrets/eks.amazonaws.com/serviceaccount/** マウントポイントに存在することを確認します。

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

出力例:

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

1 Pod ID Webhook によって Pod にマウントされた **aws-iam-token** ボリューム内の OIDC トークンファイル。トークンは、AWS でサービスアカウントの ID を認証するために使用されます。

7. Pod で、AWS Boto3 SDK オペレーションが正常に実行されることを確認します。
   a. Pod の対話型ターミナルで、Python 3 シェルを開始します。

```
$ python3
```

   b. Python 3 シェルで、**boto3** モジュールをインポートします。

```
>>> import boto3
```

   c. Boto3 **s3** サービスリソースを含む変数を作成します。

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

   d. AWS アカウントのすべての S3 バケットの名前を出力します。

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

出力例:

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

サービスアカウントが AWS IAM ロールを正常に引き受けた場合は、AWS アカウントで使用可能なすべての S3 バケットが出力に一覧表示されます。

9.4. 関連情報

- サービスアカウントで AWS IAM ロールを使用する方法の詳細は、AWS ドキュメントの サービスアカウントの IAM ロール を参照してください。
- AWS IAM ロールの委任は、AWS ドキュメントの AWS サービスにアクセス許可を委任するロールの作成 を参照してください。
- AWS SDK の詳細は、AWS ドキュメントの AWS SDKs and Tools Reference Guide を参照してください。
- AWS Boto3 SDK for Python のインストールと使用の詳細は、AWS Boto3 ドキュメント を参照してください。
- OpenShift の Webhook アドミッションプラグインに関する一般的な情報は、OpenShift Container Platform ドキュメントの Webhook アドミッションプラグイン を参照してください。

第10章 スコープトークン

10.1. トークンのスコープについて

スコープ付きトークンを作成して、パーミッションの一部を別のユーザーまたはサービスアカウントに委任できます。たとえば、プロジェクト管理者は Pod の作成権限を委任する必要があるかもしれません。

スコープ付きトークンは、指定されるユーザーを識別しますが、そのスコープによって特定のアクションに制限されるトークンです。**dedicated-admin** ロールを持つユーザーのみがスコープ付きトークンを作成できます。

スコープは、トークンの一連のスコープを **PolicyRules** のセットに変換して評価されます。次に、要求がそれらのルールに対してマッチングされます。要求属性は、追加の認可検査のために "標準" のオーソライザーに渡せるよう、スコープルールのいずれかに一致している必要があります。

10.1.1. ユーザースコープ

ユーザースコープでは、指定されたユーザーについての情報を取得することにフォーカスが置かれます。それらはインテントベースであるため、ルールは自動的に作成されます。

- **user:full**: ユーザーのすべてのパーミッションによる API の完全な読み取り/書き込みアクセスを許可します。
- **user:info**: 名前やグループなどのユーザーについての情報の読み取り専用アクセスを許可します。
- **user:check-access**: **self-localsubjectaccessreviews** および **self-subjectaccessreviews** へのアクセスを許可します。これらは、要求オブジェクトの空のユーザーおよびグループを渡す変数です。
- **user:list-projects**: ユーザーがアクセスできるプロジェクトをリスト表示するための読み取り専用アクセスを許可します。

10.1.2. ロールスコープ

ロールスコープにより、namespace でフィルターされる指定ロールと同じレベルのアクセスを持たせることができます。

- **role:<cluster-role name>:<namespace or * for all>**: 指定された namespace のみにあるクラスターロール (cluster-role) で指定されるルールにスコープを制限します。

注記

注意: これは、アクセスのエスカレートを防ぎます。ロールはシークレット、ロールバインディング、およびロールなどのリソースへのアクセスを許可しますが、このスコープはそれらのリソースへのアクセスを制限するのに役立ちます。これにより、予期しないエスカレーションを防ぐことができます。**edit** などのロールはエスカレートされるロールと見なされないことが多いですが、シークレットのアクセスを持つロールの場合はエスカレーションが生じます。

- **role:<cluster-role name>:<namespace or * for all>:!**: bang (!) を含めることでこのスコープでアクセスのエスカレートを許可されますが、それ以外には上記の例と同様になります。

第11章 バインドされたサービスアカウントトークンの使用

AWS IAM などのクラウドプロバイダーのアイデンティティーアクセス管理 (IAM) サービスとの統合を強化するバインドされたサービスアカウントトークンを使用できます。

11.1. バインドされたサービスアカウントトークンについて

バインドされたサービスアカウントトークンを使用して、所定のサービスアカウントトークンのパーミッションの範囲を制限できます。これらのトークンは対象であり、時間のバインドがあります。これにより、サービスアカウントの IAM ロールへの認証と Pod にマウントされた一時的な認証情報の生成が容易になります。ボリュームのローテーションと TokenRequest API を使用してバインドされたサービスアカウントのトークンを要求できます。

11.2. ボリュームローテーションを使用したバインドされたサービスアカウントトークンの設定

ボリュームのデプロイメントを使用してバインドされたサービスアカウントのトークンを要求するように Pod を設定できます。

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。
- サービスアカウントを作成している。この手順では、サービスアカウントの名前が **build-robot** であることを前提としています。

手順

1. ボリュームの展開を使用してバインドされたサービスアカウントのトークンを使用するように Pod を設定します。
   a. 以下の内容を含む **pod-projected-svc-token.yaml** ファイルを作成します。

```
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. 侵害のリスクを最小限に抑えるために、コンテナーが root として実行されるのを防ぎます。
2. リスクを軽減するために、必須のシステムコールに限定してデフォルトの seccomp プロファイルを設定します。
3. 既存のサービスアカウントへの参照。
4. トークンのデプロイメント先となるファイルのマウントポイントに対する相対パス。
5. オプションで、サービスアカウントトークンの有効期限を秒単位で設定します。デフォルト値は 3600 秒 (1 時間) であり、この値は少なくとも 600 秒 (10 分) である必要があります。トークンの有効期間がその 80% を過ぎている場合や、トークンの生成から 24 時間を経過している場合、kubelet はトークンのローテーションの試行を開始します。
6. オプションで、トークンの意図された対象を設定します。トークンの受信側は、受信側のアイデンティティーがトークンの適切対象の要求と一致することを確認し、一致しない場合はトークンを拒否する必要があります。対象はデフォルトで API サーバーの識別子に設定されます。

注記

予期しない障害を防ぐために、Red Hat OpenShift Service on AWS は **--service-account-extend-token-expiration** のデフォルトを **true** にして、**expirationSeconds** の値を最初のトークンの生成から 1 年になるようにオーバーライドします。この設定は変更できません。

b. Pod を作成します。

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

kubelet は Pod に代わってトークンを要求し、保存し、トークンを設定可能なファイルパスで Pod に対して利用可能にし、有効期限に達するとトークンを更新します。

2. バインドされたトークンを使用するアプリケーションは、ローテーション時にトークンのリロードを処理する必要があります。
トークンの有効期間がその 80% を過ぎている場合や、トークンの生成から 24 時間を経過している場合、kubelet はトークンをローテーションします。

11.3. POD の外部でバインドされたサービスアカウントトークンを作成する

前提条件

- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

- サービスアカウントを作成している。この手順では、サービスアカウントの名前が **build-robot** であることを前提としています。

手順

- 次のコマンドを実行して、Pod の外部にバインドされたサービスアカウントトークンを作成します。

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

出力例

```
eyJhbGciOiJSUzI1NiIsImtpZCI6IkY2M1N4MHRvc2xFNnFSQlA4eG9GYzVPdnN3NkhIV0tRW
mFrUDRNcWx4S0kifQ.eyJhdWQiOlsiaHR0cHM6Ly9pc3N1ZXIyLnRlc3QuY29tIiwiaHR0cHM6L
y9pc3N1ZXIxLnRlc3QuY29tIiwiaHR0cHM6Ly9rdWJlcm5ldGVzLmRlZmF1bHQuc3ZjIl0sImV4c
CI6MTY3OTU0MzgzMCwiaWF0IjoxNjc5NTQwMjMwLCJpc3MiOiJodHRwczovL2lzc3VlcjIudGV
zdC5jb20iLCJrdWJlcm5ldGVzLmlvIjp7Im5hbWVzcGFjZSI6ImRlZmF1bHQiLCJzZXJ2aWNlYW
Njb3VudCI6eyJuYW1lIjoidGVzdC1zYSIsInVpZCI6ImM3ZjA4MjkwLWIzOTUtNGM4NC04NjI4L
TMzMTM1NTVhNWY1OSJ9fSwibmJmIjoxNjc5NTQwMjMwLCJzdWIiOiJzeXN0ZW06c2Vydmlj
ZWFjY291bnQ6ZGVmYXVsdDp0ZXN0LXNhIn0.WyAOPvh1BFMUl3LNhBCrQeaB5wSynbnCf
ojWuNNPSilT4YvFnKibxwREwmzHpV4LO1xOFZHSi6bXBOmG_o-
m0XNDYL3FrGHd65mymiFyluztxa2lgHVxjw5reIV5ZLgNSol3Y8bJqQqmNg3rtQQWRML2kpJB
XdDHNww0E5XOypmffYkfkadli8lN5QQD-
MhsCbiAF8waCYs8bj6V6Y7uUKTcxee8sCjiRMVtXKjQtooERKm-
CH_p57wxCljIBeM89VdaR51NJGued4hVV5lxvVrYZFu89lBEAq4oyQN_d6N1vBWGXQMyoihn
t_fQjn-NfnlJWk-3NSZDIluDJAv7e-MTEk3geDrHVQKNEzDei2-Un64hSzb-
n1g1M0Vn0885wQBQAePC9UlZm8YZlMNk1tq6wIUKQTMv3HPfi5HtBRqVc2eVs0EfMX4-x-
PHhPCasJ6qLJWyj6DvyQ08dP4DW_TWZVGvKlmId0hzwpg59TTcLR0iCkISEJgAVEEd13Aa_
M0-
faD11L3MhUGxw0qxgOsPczdXUsolSISbefs7OKymzFSIkTAn9sDQ8PHMOsuyxsK8vzfrR-
E0z7MAeguZ2kaIY7cZqbN6WFy0caWgx46hrKem9vCKALefElRYbCg3hcBmowBcRTOqaFHL
NnHghhU1LaRpoFzH7OUarqX9SGQ
```

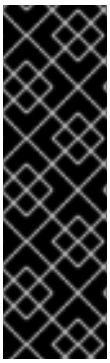
関連情報

- サービスアカウントの作成

第12章 SECURITY CONTEXT CONSTRAINTS の管理

Red Hat OpenShift Service on AWS では、SSC (Security Context Constraints) を使用して、クラスター内の Pod のアクセス許可を制御できます。

デフォルトの SCC は、インストール中、および一部の Operator またはその他のコンポーネントをインストールするときに作成されます。クラスター管理者は、OpenShift CLI (**oc**) を使用して独自の SCC を作成することもできます。



重要

デフォルトの SCC は変更しないでください。デフォルトの SCC をカスタマイズすると、一部のプラットフォーム Pod がデプロイされるか、ROSA がアップグレードされるときに問題が発生する可能性があります。さらに、一部のクラスターのアップグレード中にデフォルトの SCC 値がデフォルトにリセットされ、それらの SCC に対するすべてのカスタマイズが破棄されます。

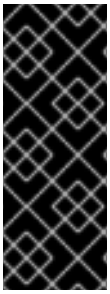
デフォルトの SCC を変更する代わりに、必要に応じて独自の SCC を作成および変更します。詳細な手順は、セキュリティーコンテキスト制約の作成 を参照してください。

12.1. SECURITY CONTEXT CONSTRAINTS について

RBAC リソースがユーザーアクセスを制御するのと同じ方法で、管理者は Security Context Constraints (SCC) を使用して Pod のパーミッションを制御できます。これらのアクセス許可によって、Pod が実行できるアクションとアクセスできるリソースが決まります。SCC を使用して、Pod がシステムに受け入れられるために必要な Pod の実行に関する条件の一覧を定義できます。

管理者は Security Context Constraints で、以下を制御できます。

- Pod が **allowPrivilegedContainer** フラグが付いた特権付きコンテナーを実行できるかどうか
- Pod が **allowPrivilegeEscalation** フラグで制約されているかどうか
- コンテナーが要求できる機能
- ホストディレクトリーのボリュームとしての使用
- コンテナーの SELinux コンテキスト
- コンテナーのユーザー ID
- ホストの namespace とネットワークの使用
- Pod ボリュームを所有する **FSGroup** の割り当て
- 許可される補助グループの設定
- コンテナーが root ファイルシステムへの書き込みアクセスを必要とするかどうか
- ボリュームタイプの使用
- 許可される **seccomp** プロファイルの設定



重要

Red Hat OpenShift Service on AWS の namespace には **openshift.io/run-level** ラベルを設定しないでください。このラベルは、Kubernetes API サーバーや OpenShift API サーバーなどのメジャー API グループの起動管理に、内部の Red Hat OpenShift Service on AWS コンポーネントで使用されます。**openshift.io/run-level** ラベルが設定される場合には対象の namespace の Pod に SCC が適用されず、その namespace で実行されるワークロードには高度な特権が割り当てられます。

12.1.1. デフォルトの Security Context Constraints

クラスターには、以下の表で説明されているように、デフォルトの SCC (Security Context Constraints) が複数含まれます。追加の SCC は、Operator または他のコンポーネントの Red Hat OpenShift Service on AWS へのインストール時にインストールされる可能性があります。

重要

デフォルトの SCC は変更しないでください。デフォルトの SCC をカスタマイズすると、一部のプラットフォーム Pod がデプロイされるか、ROSA がアップグレードされるときに問題が発生する可能性があります。さらに、一部のクラスターのアップグレード中にデフォルトの SCC 値がデフォルトにリセットされ、それらの SCC に対するすべてのカスタマイズが破棄されます。

デフォルトの SCC を変更する代わりに、必要に応じて独自の SCC を作成および変更します。詳細な手順は、**セキュリティーコンテキスト制約の作成** を参照してください。

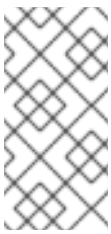
表12.1 デフォルトの Security Context Constraints

SCC (Security Context Constraints)	説明
anyuid	SCC のすべての機能が **restricted** で提供されますが、ユーザーは任意の UID と GID で実行できます。
hostaccess	ホストの全 namespace にアクセスできますが、対象の namespace に割り当てられた UID および SELinux コンテキストで Pod を実行する必要があります。 **警告** この SCC で、ホストは namespace、ファイルシステム、および PID にアクセスできます。信頼できる Pod だけがこの SCC を使用する必要があります。付与には注意が必要です。

SCC (Security Context Constraints)	説明
hostmount-anyuid	SCC のすべての機能を **restricted** で提供しますが、ホストのマウントとシステム上の任意の UID および GID として実行できます。 **警告** この SCC は、UID 0 を含む任意の UID としてホストファイルシステムにアクセスできます。付与には注意が必要です。
hostnetwork	ホストのネットワークおよびホストポートを使用できますが、対象の namespace に割り当てられた UID および SELinux コンテキストで Pod を実行する必要があります。 **警告** 追加のワークロードをコントロールプレーンホストで実行する場合は、**hostnetwork** へのアクセスを割り当てるときに注意してください。コントロールプレーンホストで **hostnetwork** を実行するワークロードにはクラスター上で root アクセスを持つユーザーと同じ機能があるため、適切に信頼されている必要があります。
hostnetwork-v2	**hostnetwork** SCC と似ていますが、次の違いがあります。 • **ALL** 機能がコンテナーから削除されます。 • **NET_BIND_SERVICE** 機能を明示的に追加できます。 • **seccompProfile** はデフォルトで **runtime/default** に設定されています。 • セキュリティーコンテキストでは、**allowPrivilegeEscalation** を設定解除するか、**false** に設定する必要があります。

SCC (Security Context Constraints)	説明
node-exporter	Prometheus ノードエクスポーターに使用されます。 **警告** この SCC は、UID 0 を含む任意の UID としてホストファイルシステムにアクセスできます。付与には注意が必要です。
nonroot	SCC のすべての機能が **restricted** で提供されますが、ユーザーは root 以外の UID で実行できます。ユーザーは UID を指定するか、コンテナーランタイムのマニフェストに指定する必要があります。
nonroot-v2	**nonroot** SCC と同様ですが、次の違いがあります。 • **ALL** 機能がコンテナーから削除されます。 • **NET_BIND_SERVICE** 機能を明示的に追加できます。 • **seccompProfile** はデフォルトで **runtime/default** に設定されています。 • セキュリティーコンテキストでは、**allowPrivilegeEscalation** を設定解除するか、**false** に設定する必要があります。

SCC (Security Context Constraints)	説明
privileged	すべての特権およびホスト機能にアクセスでき、任意のユーザー、任意のグループ、FSGroup、および任意の SELinux コンテキストで実行できます。 **警告** これは最も制限の少ない SCC であり、クラスター管理にのみ使用してください。付与には注意が必要です。 **privileged** SCC は以下を許可します。 ● ユーザーによる特権付き Pod の実行 ● Pod によるホストディレクトリーのボリュームとしてのマウント ● Pod の任意ユーザーとしての実行 ● Pod の MCS ラベルの使用による実行 ● Pod によるホストの IPC namespace の使用 ● Pod によるホストの PID namespace の使用 ● Pod による FSGroup の使用 ● Pod による補助グループの使用 ● Pod による seccomp プロファイルの使用 ● Pod による機能の要求 **注記** Pod の仕様で **privileged: true** を設定しても、**privileged** SCC が選択されるとは限りません。ユーザーに使用権限がある場合に、**allowPrivilegedContainer: true** が指定されており、優先順位が最も高い SCC が選択されます。

SCC (Security Context Constraints)	説明
restricted	すべてのホスト機能へのアクセスが拒否され、Pod を UID および namespace に割り当てられる SELinux コンテキストで実行する必要があります。 **restricted** SCC は以下を実行します。 ● Pod が特権付きで実行されないようにします。 ● Pod がホストディレクトリーボリュームをマウントできないようにします。 ● Pod が事前に割り当てられた UID 範囲のユーザーとして実行することを要求します。 ● Pod が事前に割り当てられた MCS ラベルで実行されることを要求します。 ● Pod が事前に割り当てられた FSGroup で実行されることを要求します。 ● Pod が補助グループを使用することを許可します。 Red Hat OpenShift Service on AWS 4.10 以前からアップグレードされたクラスターでは、すべての認証済みユーザーがこの SCC を使用できます。アクセスが明示的に付与されない限り、新しい Red Hat OpenShift Service on AWS 4.11 以降のインストールのユーザーは **restricted** SCC を使用できなくなります。
restricted-v2	**restricted** SCC と同様ですが、次の違いがあります。 ● **ALL** 機能がコンテナーから削除されます。 ● **NET_BIND_SERVICE** 機能を明示的に追加できます。 ● **seccompProfile** はデフォルトで **runtime/default** に設定されています。 ● セキュリティーコンテキストでは、**allowPrivilegeEscalation** を設定解除するか、**false** に設定する必要があります。 これは新規インストールで提供され、デフォルトで認証済みユーザーに使用される最も制限の厳しい SCC です。  **注記** **restricted-v2** SCC は、システムにデフォルトで含まれている SCC の中で最も制限が厳しいものです。ただし、さらに制限の厳しいカスタム SCC を作成できます。たとえば、**readOnlyRootFilesystem** を **true** に制限する SCC を作成できます。

12.1.2. Security Context Constraints の設定

Security Context Constraints (SCC) は、Pod がアクセスできるセキュリティー機能を制御する設定およびストラテジーで設定されています。これらの設定は以下のカテゴリーに分類されます。

カテゴリー	説明
ブール値による制御	このタイプのフィールドはデフォルトで最も制限のある値に設定されます。たとえば、**AllowPrivilegedContainer** が指定されていない場合は、**false** に常に設定されます。
許可されるセットによる制御	このタイプのフィールドがセットに対してチェックされ、その値が許可されることを確認します。
ストラテジーによる制御	値を生成するストラテジーを持つ項目は以下を提供します。 ● 値を生成するメカニズム ● 指定された値が許可される値のセットに属するようにするメカニズム

CRI-O には、Pod の各コンテナーについて許可されるデフォルトの機能リストがあります。

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

コンテナーはこのデフォルトリストから機能を使用しますが、Pod マニフェストの作成者は追加機能を要求したり、デフォルト動作の一部を削除してリストを変更できます。**allowedCapabilities** パラメーター、**defaultAddCapabilities** パラメーター、および **requiredDropCapabilities** パラメーターを使用して、Pod からのこのような要求を制御します。これらのパラメーターを使用して、(各コンテナーに追加する必要のある機能や、各コンテナーから禁止または破棄する必要のあるものなど) 要求できる機能を指定できます。

注記

requiredDropCapabilities パラメーターを **ALL** に設定すると、すべての capabilites をコンテナーから取り除くことができます。これは、**restricted-v2** SCC の機能です。

12.1.3. Security Context Constraints ストラテジー

RunAsUser

- **MustRunAs**: **runAsUser** が設定されることを要求します。デフォルトで設定済みの **runAsUser** を使用します。設定済みの **runAsUser** に対して検証します。

MustRunAs スニペットの例

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

- **MustRunAsRange**: 事前に割り当てられた値を使用していない場合に、最小値および最大値が定義されることを要求します。デフォルトでは最小値を使用します。許可される範囲全体に対して検証します。

  MustRunAsRange スニペットの例

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

- **MustRunAsNonRoot**: Pod がゼロ以外の **runAsUser** で送信されること、または **USER** ディレクティブをイメージに定義することを要求します。デフォルトは指定されません。

  MustRunAsNonRoot スニペットの例

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

- **RunAsAny**: デフォルトは指定されません。**runAsUser** の指定を許可します。

  RunAsAny スニペットの例

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

SELinuxContext

- **MustRunAs**: 事前に割り当てられた値を使用していない場合に **seLinuxOptions** が設定されることを要求します。デフォルトとして **seLinuxOptions** を使用します。**seLinuxOptions** に対して検証します。
- **RunAsAny**: デフォルトは指定されません。**seLinuxOptions** の指定を許可します。

SupplementalGroups

- **MustRunAs**: 事前に割り当てられた値を使用していない場合に、少なくとも 1 つの範囲が指定されることを要求します。デフォルトとして最初の範囲の最小値を使用します。すべての範囲に対して検証します。

- **RunAsAny**: デフォルトは指定されません。**supplementalGroups** の指定を許可します。

FSGroup

- **MustRunAs**: 事前に割り当てられた値を使用していない場合に、少なくとも 1 つの範囲が指定されることを要求します。デフォルトとして最初の範囲の最小値を使用します。最初の範囲の最初の ID に対して検証します。
- **RunAsAny**: デフォルトは指定されません。**fsGroup** ID の指定を許可します。

12.1.4. ボリュームの制御

特定のボリュームタイプの使用は、SCC の **volumes** フィールドを設定して制御できます。

このフィールドの許容値は、ボリュームの作成時に定義されるボリュームソースに対応します。

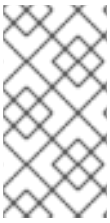
- **awsElasticBlockStore**
- **azureDisk**
- **azureFile**
- **cephFS**
- **cinder**
- **configMap**
- **csi**
- **downwardAPI**
- **emptyDir**
- **fc**
- **flexVolume**
- **flocker**
- **gcePersistentDisk**
- **ephemeral**
- **gitRepo**
- **glusterfs**
- **hostPath**
- **iscsi**
- **nfs**
- **persistentVolumeClaim**
- **photonPersistentDisk**

- **portworxVolume**
- **projected**
- **quobyte**
- **rbd**
- **scaleIO**
- **secret**
- **storageos**
- **vsphereVolume**
- * (すべてのボリュームタイプの使用を許可する特殊な値)
- **none** (すべてのボリュームタイプの使用を無効にする特殊な値。後方互換の場合にのみ存在する)

新規 SCC について許可されるボリュームの推奨される最小セットは、**configMap**、**downwardAPI**、**emptyDir**、**persistentVolumeClaim**、**secret**、および **projected** です。

注記

Red Hat OpenShift Service on AWS の各リリースに新しいタイプ追加されるため、この許可されるボリュームタイプリストがすべて網羅しているわけではありません。



注記

後方互換性を確保するため、**allowHostDirVolumePlugin** の使用は **volumes** フィールドの設定をオーバーライドします。たとえば、**allowHostDirVolumePlugin** が false に設定されていて、**volumes** フィールドで許可されている場合は、**volumes** から **hostPath** 値が削除されます。

12.1.5. 受付制御

SCC が設定された **受付制御** により、ユーザーに付与された機能に基づいてリソースの作成に対する制御が可能になります。

SCC の観点では、これは受付コントローラーが、SCC の適切なセットを取得するためにコンテキストで利用可能なユーザー情報を検査できることを意味します。これにより、Pod はその運用環境についての要求を行ったり、Pod に適用する一連の制約を生成したりする権限が与えられます

受付が Pod を許可するために使用する SCC のセットはユーザーアイデンティティーおよびユーザーが属するグループによって決定されます。さらに、Pod がサービスアカウントを指定する場合は、許可される SCC のセットに、サービスアカウントでアクセスできる制約が含まれます。

注記

デプロイメントなどのワークロードリソースを作成する場合、SCC の検索と、作成された Pod の許可には、サービスアカウントのみが使用されます。

受付は以下の方法を使用して、Pod の最終的なセキュリティーコンテキストを作成します。

1. 使用できるすべての SCC を取得します。
2. 要求に指定されていないセキュリティーコンテキストに、設定のフィールド値を生成します。
3. 利用可能な制約に対する最終的な設定を検証します。

制約の一致するセットが検出される場合は、Pod が受け入れられます。要求が SCC に一致しない場合は、Pod が拒否されます。

Pod はすべてのフィールドを SCC に対して検証する必要があります。以下は、検証する必要のある 2 つのフィールドのみについての例になります。

注記

これらの例は、事前に割り当てられた値を使用するストラテジーに関連します。

MustRunAs の FSGroup SCC ストラテジー

Pod が **fsGroup** ID を定義する場合、その ID はデフォルトの **fsGroup** ID に等しくなければなりません。そうでない場合は、Pod が SCC で検証されず、次の SCC が評価されます。

SecurityContextConstraints.fsGroup フィールドに値 **RunAsAny** があり、Pod 仕様が **Pod.spec.securityContext.fsGroup** を省略すると、このフィールドは有効とみなされます。検証時に、他の SCC 設定が他の Pod フィールドを拒否し、そのため Pod を失敗させる可能性があることに注意してください。

MustRunAs の SupplementalGroups SCC ストラテジー

Pod 仕様が 1 つ以上の **supplementalGroups** ID を定義する場合、Pod の ID は namespace の **openshift.io/sa.scc.supplemental-groups** アノテーションの ID のいずれかに等しくなければなりません。そうでない場合は、Pod が SCC で検証されず、次の SCC が評価されます。

SecurityContextConstraints.supplementalGroups フィールドに値 **RunAsAny** があり、Pod 仕様が **Pod.spec.securityContext.supplementalGroups** を省略する場合、このフィールドは有効とみなされます。検証時に、他の SCC 設定が他の Pod フィールドを拒否し、そのため Pod を失敗させる可能性があることに注意してください。

12.1.6. Security Context Constraints の優先度設定

SCC (Security Context Constraints) には優先度フィールドがあり、受付コントローラーの要求検証を試行する順序に影響を与えます。

優先順位値 **0** は可能な限り低い優先順位です。nil 優先順位は **0** または最低の優先順位と見なされます。優先順位の高い SCC は、並べ替え時にセットの先頭に移動します。

使用可能な SCC の完全なセットが決定すると、SCC は次の方法で順序付けられます。

1. 最も優先度の高い SCC が最初に並べられます。
2. 優先度が等しい場合、SCC は最も制限の多いものから少ないものの順に並べ替えられます。
3. 優先度と制限の両方が等しい場合、SCC は名前でソートされます。

デフォルトで、クラスター管理者に付与される **anyuid** SCC には SCC セットの優先度が指定されます。これにより、クラスター管理者は Pod の **SecurityContext** で **RunAsUser** を指定することにより、任意のユーザーとして Pod を実行できます。

12.2. 事前に割り当てられる SECURITY CONTEXT CONSTRAINTS 値について

受付コントローラーは、これが namespace の事前に割り当てられた値を検索し、Pod の処理前に Security Context Constraints (SCC) を設定するようにトリガーする SCC (Security Context Constraint) の特定の条件を認識します。各 SCC ストラテジーは他のストラテジーとは別に評価されます。この際、(許可される場合に) Pod 仕様の値と共に集計された各ポリシーの事前に割り当てられた値が使用され、実行中の Pod で定義される各種 ID の最終の値が設定されます。

以下の SCC により、受付コントローラーは、範囲が Pod 仕様で定義されていない場合に事前に定義された値を検索できます。

1. 最小または最大値が設定されていない **MustRunAsRange** の **RunAsUser** ストラテジーです。受付は **openshift.io/sa.scc.uid-range** アノテーションを検索して範囲フィールドを設定します。

2. レベルが設定されていない **MustRunAs** の **SELinuxContext** ストラテジーです。受付は **openshift.io/sa.scc.mcs** アノテーションを検索してレベルを設定します。

3. **MustRunAs** の **FSGroup** ストラテジーです。受付は、**openshift.io/sa.scc.supplemental-groups** アノテーションを検索します。

4. **MustRunAs** の **SupplementalGroups** ストラテジーです。受付は、**openshift.io/sa.scc.supplemental-groups** アノテーションを検索します。

生成フェーズでは、セキュリティーコンテキストのプロバイダーが Pod にとくに設定されていないパラメーター値をデフォルト設定します。デフォルト設定は選択されるストラテジーに基づいて行われます。

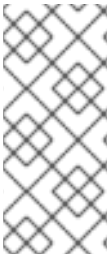
1. **RunAsAny** および **MustRunAsNonRoot** ストラテジーはデフォルトの値を提供しません。Pod がパラメーター値 (グループ ID など) を必要とする場合は、値を Pod 仕様内に定義する必要があります。

2. **MustRunAs** (単一の値) ストラテジーは、常に使用されるデフォルト値を提供します。たとえば、グループ ID の場合は、Pod 仕様が独自の ID 値を定義する場合でも、namespace のデフォルトパラメーター値が Pod のグループに表示されます。

3. **MustRunAsRange** および **MustRunAs** (範囲ベース) ストラテジーは、範囲の最小値を提供します。単一値の **MustRunAs** ストラテジーの場合のように、namespace のデフォルト値は実行中の Pod に表示されます。範囲ベースのストラテジーが複数の範囲で設定可能な場合、これは最初に設定された範囲の最小値を指定します。

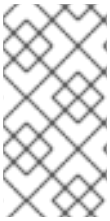
注記

FSGroup および **SupplementalGroups** ストラテジーは、**openshift.io/sa.scc.supplemental-groups** アノテーションが namespace に存在しない場合に **openshift.io/sa.scc.uid-range** アノテーションにフォールバックします。いずれも存在しない場合は、SCC が作成されません。



注記

デフォルトで、アノテーションベースの **FSGroup** ストラテジーは、自身をアノテーションの最小値に基づく単一の範囲で設定します。たとえば、アノテーションが **1/3** を読み取ると、**FSGroup** ストラテジーは **1** の最小値および最大値で自身を設定します。追加のグループを **FSGroup** フィールドで許可する必要がある場合は、アノテーションを使用しないカスタム SCC を設定することができます。



注記

openshift.io/sa.scc.supplemental-groups アノテーションは、**<start>/<length** または **<start>-<end>** 形式のコンマ区切りのブロックのリストを受け入れます。**openshift.io/sa.scc.uid-range** アノテーションは単一ブロックのみを受け入れます。

12.3. SECURITY CONTEXT CONSTRAINTS の例

以下の例は、Security Context Constraints (SCC) 形式およびアノテーションを示しています。

アノテーション付き 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 が要求できる機能の一覧です。特殊な記号 * は任意の機能を許可しますが、リストが空の場合は、いずれの機能も要求できないことを意味します。
2. Pod に含める追加機能のリストです。
3. セキュリティーコンテキストの許可される値を定める **FSGroup** ストラテジータイプです。
4. この SCC へのアクセスを持つグループです。
5. Pod から取り除く機能のリストです。または、**ALL** を指定してすべての機能をドロップします。
6. セキュリティーコンテキストの許可される値を定める **runAsUser** ストラテジータイプです。
7. セキュリティーコンテキストの許可される値を定める **seLinuxContext** ストラテジータイプです。
8. セキュリティーコンテキストの許可される補助グループを定める **supplementalGroups** ストラテジーです。
9. この SCC にアクセスできるユーザーです。
10. セキュリティーコンテキストで許容されるボリュームタイプです。この例では、* はすべてのボリュームタイプの使用を許可します。

SCC の **users** フィールドおよび **groups** フィールドは SCC にアクセスできるユーザー制御します。デフォルトで、クラスター管理者、ノードおよびビルドコントローラーに特権付き SCC へのアクセスが付与されます。認証されたすべてのユーザーには **restricted-v2** SCC へのアクセスが付与されます。

明示的な runAsUser 設定を使用しない場合

```
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 コンテナーまたは Pod が実行時に使用するユーザー ID を要求しない場合、有効な UID はこの Pod を作成する SCC よって異なります。**restricted-v2** SCC はデフォルトですべての認証ユーザーに付与されるため、ほとんどの場合はすべてのユーザーおよびサービスアカウントで利用でき、使用されます。**restricted-v2** SCC は、**securityContext.runAsUser** フィールドの使用できる値を制限し、これをデフォルトに設定するために **MustRunAsRange** ストラテジーを使用します。受付プラグインではこの範囲を指定しないため、現行プロジェクトで **openshift.io/sa.scc.uid-range** アノテーションを検索して範囲フィールドにデータを設定します。最終的にコンテナーの **runAsUser** は予測が困難な範囲の最初の値と等しい値になります。予測が困難であるのはすべてのプロジェクトにはそれぞれ異なる範囲が設定されるためです。

明示的な runAsUser 設定を使用する場合

```
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 特定のユーザー ID を要求するコンテナーまたは Pod が Red Hat OpenShift Container Platform on AWS で受け入れられるのは、サービスアカウントまたはユーザーに、そのユーザー ID を許可するように、SCC へのアクセスが付与されている場合のみです。SCC は、任意の ID や特定の範囲内にある ID、または要求に固有のユーザー ID を許可します。

この設定は、SELinux、fsGroup、および Supplemental Groups について有効です。

12.4. セキュリティーコンテキスト制約の作成

OpenShift CLI (**oc**) を使用して Security Context Constraints (SCC) を作成できます。

重要

独自の SCC の作成と変更は高度な操作であり、クラスターを不安定にする可能性があります。独自の SCC の使用について質問がある場合は、Red Hat サポートにお問い合わせください。Red Hat サポートへの連絡方法は、**サポートを受ける方法** を参照してください。

前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** ロールを持つユーザーとしてクラスターにログインしている。

手順

1. **scc-admin.yaml** という名前の YAML ファイルで SCC を定義します。

```
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
```

オプションとして、**requiredDropCapabilities** フィールドに必要な値を設定して、SCC の特定の機能を取り除くことができます。指定された機能はコンテナーからドロップされます。すべてのケイパビリティーを破棄するには、**ALL** を指定します。たとえば、**KILL** 機能、**MKNOD** 機能、および **SYS_CHROOT** 機能のない SCC を作成するには、以下を SCC オブジェクトに追加します。

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

注記

allowedCapabilities と **requiredDropCapabilities** の両方に、機能を追加できません。

CRI-O は、Docker ドキュメント に記載されている同じ一連の機能の値をサポートし ます。

2. ファイルを渡して SCC を作成します。

```
$ oc create -f scc-admin.yaml
```

出力例

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

検証

- SCC が作成されていることを確認します。

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

出力例

```
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. 特定の SCC を必要とするワークロードの設定

特定のセキュリティーコンテキスト制約 (SCC) を要求するようにワークロードを設定できます。これは、特定の SCC をワークロードに固定する場合、または必要な SCC がクラスター内の別の SCC によってプリエンプションされるのを防ぐ場合に役立ちます。

特定の SCC を要求するには、ワークロードに **openshift.io/required-scc** アノテーションを設定します。このアノテーションは、デプロイメントやデーモンセットなど、Pod マニフェストテンプレートを設定できる任意のリソースに設定できます。

SCC はクラスター内に存在し、ワークロードに適用できる必要があります。そうでない場合、Pod のアドミッションは失敗します。Pod を作成するユーザーまたは Pod のサービスアカウントが Pod の namespace で SCC の **use** 権限を持っている場合、SCC はワークロードに適用可能であるとみなされます。

警告

ライブ Pod のマニフェスト内の **openshift.io/required-scc** アノテーションを変更しないでください。変更すると、Pod のアドミッションが失敗するためです。必要な SCC を変更するには、基礎となる Pod テンプレートのアノテーションを更新します。これにより、Pod が削除され、再作成されます。

前提条件

- SCC はクラスター内に存在する必要があります。

手順

1. デプロイメント用の YAML ファイルを作成し、**openshift.io/required-scc** アノテーションを設定して必要な SCC を指定します。

   deployment.yaml の例

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

1 必要な SCC の名前を指定します。

2. 次のコマンドを実行して、リソースを作成します。

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

検証

- デプロイメントで指定された SCC が使用されたことを確認します。
  a. 次のコマンドを実行して、Pod の **openshift.io/scc** アノテーションの値を表示します。

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

1 **<pod_name>** をデプロイメント Pod の名前に置き換えます。

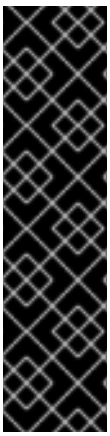
  b. 出力を調べて、表示された SCC がデプロイメントで定義した SCC と一致することを確認します。

出力例

```
my-scc
```

12.6. SECURITY CONTEXT CONSTRAINTS へのロールベースのアクセス

SCC は RBAC で処理されるリソースとして指定できます。これにより、SCC へのアクセスのスコープを特定プロジェクトまたはクラスター全体に設定できます。ユーザー、グループ、またはサービスアカウントを SCC に直接割り当てると、クラスター全体のスコープが保持されます。



重要

デフォルトプロジェクトでワークロードを実行したり、デフォルトプロジェクトへのアクセスを共有したりしないでください。デフォルトのプロジェクトは、コアクラスターコンポーネントを実行するために予約されています。

次のデフォルトプロジェクトは、高い特権があるとみなされます (**default**、**kube-public**、**kube-system**、**openshift**、**openshift-infra**、**openshift-node**、および **openshift.io/run-level** ラベルが **0** または **1** に設定されているその他のシステム作成プロジェクト)。Pod セキュリティーアドミッション、セキュリティーコンテキスト制約、クラスターリソースクォータ、イメージ参照解決などのアドミッションプラグインに依存する機能は、高い特権を持つプロジェクトでは機能しません。

ロールの SCC へのアクセスを組み込むには、ロールの作成時に **scc** リソースを指定します。

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

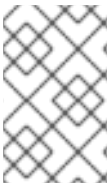
これにより、以下のロール定義が生成されます。

```
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. ロールの名前。
2. 定義されたロールの namespace。指定されていない場合は、**default** にデフォルト設定されます。
3. **SecurityContextConstraints** リソースを含む API グループ。**scc** がリソースとして指定される場合に自動的に定義されます。
4. アクセスできる SCC の名前のサンプル。
5. ユーザーが SCC 名を **resourceNames** フィールドに指定することを許可するリソースグループの名前。
6. ロールに適用する動詞のリスト。

このようなルールを持つローカルまたはクラスターロールは、ロールバインディングまたはクラスターロールバインディングでこれにバインドされたサブジェクトが **scc-name** というユーザー定義の SCC を使用することを許可します。



注記

RBAC はエスカレーションを防ぐように設計されているため、プロジェクト管理者であっても SCC へのアクセスを付与することはできません。デフォルトでは、**restricted-v2** SCC を含め、SCC リソースで動詞 **use** を使用することは許可されていません。

12.7. SECURITY CONTEXT CONSTRAINTS コマンドのリファレンス

OpenShift CLI (**oc**) を使用して、インスタンスの Security Context Constraints (SCC) を通常の API オブジェクトとして管理できます。

12.7.1. Security Context Constraints の表示

SCC の現在の一覧を取得するには、以下を実行します。

```
$ oc get scc
```

出力例

```
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. Security Context Constraints の検証

特定の SCC に関する情報 (SCC が適用されるユーザー、サービスアカウントおよびグループを含む) を表示できます。

たとえば、**restricted** SCC を検査するには、以下を実行します。

```
$ oc describe scc restricted
```

出力例

```
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. SCC が適用されるユーザーとサービスアカウントをリスト表示します。
2. SCC が適用されるグループをリスト表示します。

12.8. 関連情報

- サポート

第13章 POD セキュリティーアドミッションの理解と管理

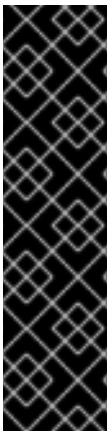
Pod セキュリティーアドミッションは、Kubernetes Pod セキュリティー標準 の実装です。Pod のセキュリティーアドミッションを使用して、Pod の動作を制限します。

13.1. POD セキュリティーアドミッションについて

Red Hat OpenShift Service on AWS には、Kubernetes Pod のセキュリティーアドミッション が組み込まれています。グローバルまたは namespace レベルで定義された Pod のセキュリティーアドミッションに準拠していない Pod は、クラスターへの参加が許可されず、実行できません。

グローバルに、**privileged** プロファイルが適用され、**restricted** プロファイルが警告と監査に使用されます。

Pod のセキュリティーアドミッション設定を namespace レベルで設定することもできます。



重要

デフォルトプロジェクトでワークロードを実行したり、デフォルトプロジェクトへのアクセスを共有したりしないでください。デフォルトのプロジェクトは、コアクラスターコンポーネントを実行するために予約されています。

次のデフォルトプロジェクトは、高い特権があるとみなされます (**default**、**kube-public**、**kube-system**、**openshift**、**openshift-infra**、**openshift-node**、および **openshift.io/run-level** ラベルが **0** または **1** に設定されているその他のシステム作成プロジェクト)。Pod セキュリティーアドミッション、セキュリティーコンテキスト制約、クラスターリソースクォータ、イメージ参照解決などのアドミッションプラグインに依存する機能は、高い特権を持つプロジェクトでは機能しません。

13.1.1. Pod のセキュリティーアドミッションモード

namespace に対して次の Pod セキュリティーアドミッションモードを設定できます。

表13.1 Pod のセキュリティーアドミッションモード

モード	ラベル	説明
enforce	**pod-security.kubernetes.io/enforce**	設定されたプロファイルに準拠していない Pod の受け入れを拒否します。
audit	**pod-security.kubernetes.io/audit**	Pod が設定されたプロファイルに準拠していない場合、監査イベントをログに記録します。
warn	**pod-security.kubernetes.io/warn**	Pod が設定されたプロファイルに準拠していない場合に警告を表示します。

13.1.2. Pod のセキュリティーアドミッションプロファイル

各 Pod セキュリティーアドミッションモードを次のプロファイルのいずれかに設定できます。

表13.2 Pod のセキュリティーアドミッションプロファイル

プロファイル	説明
privileged	最も制限の少ないポリシー。既知の権限昇格が可能になる
baseline	最小限の制限ポリシー。既知の権限昇格を防止する
restricted	最も制限的なポリシー。現在の Pod 強化のベストプラクティスに従う

13.1.3. 特権付きの namespace

次のシステム namespace は、常に **privileged** Pod セキュリティーアドミッションプロファイルに設定されます。

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

これらの特権付き namespace の Pod セキュリティープロファイルを変更することはできません。

13.1.4. Pod セキュリティーアドミッションおよびセキュリティーコンテキストの制約

Pod セキュリティーアドミッションの標準とセキュリティーコンテキストの制約は、2 つの独立したコントローラーによって調整され、適用されます。2 つのコントローラーは独立して機能し、以下のプロセスを使用してセキュリティーポリシーを適用します。

1. セキュリティーコンテキスト制約のコントローラーは、Pod に割り当てられた SCC (セキュリティーコンテキスト制約) ごとに、一部のセキュリティーコンテキストフィールドを変更する可能性があります。たとえば seccomp プロファイルが空、または設定されていない場合で、Pod に割り当てられた SCC が **seccompProfiles** フィールドを **runtime/default** に強制する場合、コントローラーはデフォルト型を **RuntimeDefault** に設定します。
2. セキュリティーコンテキスト制約のコントローラーは、一致する SCC に対して Pod のセキュリティーコンテキストを検証します。
3. Pod セキュリティーアドミッションのコントローラーは、namespace に割り当てられた Pod セキュリティー標準に対して Pod のセキュリティーコンテキストを検証します。

13.2. POD セキュリティーアドミッション同期について

グローバル Pod セキュリティーアドミッションコントロール設定に加えて、コントローラーは、特定の namespace にあるサービスアカウントの SCC アクセス許可に従って、Pod セキュリティーアドミッションコントロールの **warn** および **audit** ラベルを namespace に適用します。

コントローラーは **ServiceAccount** オブジェクトのアクセス許可を確認して、各 namespace でセキュリティーコンテキストの制約を使用します。セキュリティーコンテキスト制約 (SCC) は、フィールド値に基づいて Pod セキュリティープロファイルにマップされます。コントローラーはこれらの変換されたプロファイルを使用します。Pod のセキュリティー許可 **warn** と **audit** ラベルは、Pod の作成時に警告が表示されたり、監査イベントが記録されたりするのを防ぐために、namespace で最も特権のある Pod セキュリティープロファイルに設定されます。

namespace のラベル付けは、namespace ローカルサービスアカウントの権限を考慮して行われます。

Pod を直接適用すると、Pod を実行するユーザーの SCC 権限が使用される場合があります。ただし、自動ラベル付けではユーザー権限は考慮されません。

13.2.1. Pod セキュリティーアドミッション同期の namespace の除外

Pod セキュリティーアドミッション同期は、システムで作成された namespace および **openshift-*** 接頭辞が付いた namespace では永続的に無効になります。

クラスターペイロードの一部として定義されている namespace では、Pod セキュリティーアドミッションの同期が完全に無効になっています。次の namespace は永続的に無効になります。

- **default**
- **kube-node-lease**
- **kube-system**
- **kube-public**
- **openshift**
- **openshift-** という接頭辞が付いた、システムによって作成されたすべての namespace

13.3. POD セキュリティーアドミッションの同期制御

ほとんどの namespace で自動 Pod セキュリティーアドミッションの同期を有効または無効にできます。

重要

システム作成の namespace では、Pod セキュリティーアドミッション同期を有効にすることはできません。詳細は、**Pod のセキュリティーアドミッション同期 namespace の除外** を参照してください。

手順

- 設定する namespace ごとに、**security.openshift.io/scc.podSecurityLabelSync** ラベルの値を設定します。
  - namespace で Pod セキュリティーアドミッションラベルの同期を無効にするには、**security.openshift.io/scc.podSecurityLabelSync** ラベルの値を **false** に設定します。以下のコマンドを実行します。

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

  - namespace で Pod セキュリティーアドミッションラベルの同期を有効にするには、**security.openshift.io/scc.podSecurityLabelSync** ラベルの値を **true** に設定します。以下のコマンドを実行します。

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

関連情報

- Pod セキュリティーアドミッション同期の namespace の除外

13.4. NAMESPACE の POD セキュリティーアドミッションの設定

Pod のセキュリティーアドミッション設定を namespace レベルで設定できます。namespace の Pod セキュリティーアドミッションモードごとに、どの Pod セキュリティーアドミッションプロファイルを使用するかを設定できます。

手順

- namespace に設定する Pod セキュリティーアドミッションモードごとに、次のコマンドを実行します。

```
$ oc label namespace <namespace> \ 1
    pod-security.kubernetes.io/<mode>=<profile> \ 2
    --overwrite
```

1 **<namespace>** には、設定する namespace を指定します。

2 **<mode>** を **enforce**、**warn**、または **audit** に設定します。**<profile>** を **restricted**、**baseline**、または **privileged** に設定します。

13.5. POD セキュリティーアドミッションアラート

PodSecurityViolation アラートがトリガーされるのは、Pod セキュリティーアドミッションコントローラーの監査レベルで Pod が拒否されたことを Kubernetes API サーバーが報告された場合です。このアラートは 1 日間持続します。

Kubernetes API サーバーの監査ログを表示して、トリガーされたアラートを調査します。たとえば、グローバル適用の Pod セキュリティーレベルが **restricted** に設定されている場合には、ワークロードは承認に失敗する可能性があります。

Pod セキュリティーアドミッション違反の監査イベントを特定する方法については、Kubernetes ドキュメントの 監査アノテーション を参照してください。

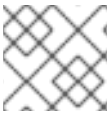
13.6. 関連情報

- 監査ログの表示
- Security Context Constraints の管理

第14章 LDAP グループの同期

dedicated-admin ロールを持つ管理者は、グループを使用して、ユーザーの管理、権限の変更、連携の強化を行うことができます。組織ではユーザーグループをすでに作成し、それらを LDAP サーバーに保存している場合があります。Red Hat OpenShift Service on AWS は、これらの LDAP レコードを内部 Red Hat OpenShift Service on AWS レコードと同期できるため、管理者はグループを 1 カ所で管理できます。Red Hat OpenShift Service on AWS は現在、グループメンバーシップを定義するための 3 つの共通スキーマ (RFC 2307、Active Directory、拡張された Active Directory) を使用したグループと LDAP サーバーの同期をサポートしています。

LDAP の設定の詳細は、LDAP アイデンティティープロバイダーの設定 を参照してください。



注記

グループを同期するには、**dedicated-admin** 権限が必要です。

14.1. LDAP 同期の設定について

LDAP 同期を実行するには、同期設定ファイルが必要です。このファイルには、以下の LDAP クライアント設定の詳細が含まれます。

- LDAP サーバーへの接続の設定。
- LDAP サーバーで使用されるスキーマに依存する同期設定オプション。
- Red Hat OpenShift Service on AWS のグループ名を LDAP サーバー内のグループにマッピングする、管理者が定義した名前マッピングのリスト。

設定ファイルの形式は、使用するスキーマ (RFC 2307、Active Directory、または拡張 Active Directory) によって異なります。

LDAP クライアント設定

設定の LDAP クライアント設定セクションでは、LDAP サーバーへの接続を定義します。

設定の LDAP クライアント設定セクションでは、LDAP サーバーへの接続を定義します。

LDAP クライアント設定

```
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) データベースをホストする LDAP サーバーの接続プロトコル、IP アドレス、および **scheme://host:port** としてフォーマットされる接続先のポートです。

(2) バインド DN として使用する任意の識別名 (DN) です。これは、同期操作のエントリーを取得するために昇格された特権が必要な場合、Red Hat OpenShift Service on AWS で使用されます。

(3) バインドに使用する任意のパスワードです。これは、同期操作のエントリーを取得するために昇格された特権が必要な場合、Red Hat OpenShift Service on AWS で使用されます。この値は環境変数、外部ファイル、または暗号化されたファイルでも指定できます。

4 **false** の場合、セキュアな LDAP (**ldaps://**) URL は TLS を使用して接続し、非セキュアな LDAP (**ldap://**) URL は TLS にアップグレードされます。 **true** の場合、サーバーへの TLS 接続は行われま

5 設定された URL のサーバー証明書を検証するために使用する証明書バンドルです。空の場合、Red Hat OpenShift Service on AWS はシステムで信頼されたルートを使用します。**insecure** が **false** に設定されている場合にのみ、これが適用されます。

LDAP クエリー定義

同期設定は、同期に必要となるエントリーの LDAP クエリー定義で設定されています。LDAP クエリーの特定の定義は、LDAP サーバーにメンバーシップ情報を保存するために使用されるスキーマに依存します。

LDAP クエリー定義

```
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 すべての検索が開始されるディレクトリーのブランチの識別名 (DN) です。ディレクトリーツリーの上部を指定する必要がありますが、ディレクトリーのサブツリーを指定することもできます。

2 検索の範囲です。有効な値は **base**、**one**、または **sub** です。これを定義しない場合、**sub** の範囲が使用されます。範囲オプションについては、以下の表で説明されています。

3 LDAP ツリーのエイリアスに関連する検索の動作です。有効な値は **never**、**search**、**base**、または **always** です。これを定義しない場合、デフォルトは **always** となり、エイリアスを逆参照します。逆参照の動作については以下の表で説明されています。

4 クライアントによって検索に許可される時間制限です。**0** の値はクライアント側の制限がないことを意味します。

5 有効な LDAP 検索フィルターです。これを定義しない場合、デフォルトは **(objectClass=*)** になります。

6 LDAP エントリーで測定される、サーバーからの応答ページの任意の最大サイズです。**0** に設定すると、応答ページのサイズ制限はなくなります。クライアントまたはサーバーがデフォルトで許可しているエントリー数より多いエントリーをクエリーが返す場合、ページングサイズの設定が必要となります。

表14.1 LDAP 検索範囲オプション

LDAP 検索範囲	説明
base	クエリーに対して指定されるベース DN で指定するオブジェクトのみを考慮します。
one	クエリーについてベース DN とツリー内の同じレベルにあるすべてのオブジェクトを考慮します。

LDAP 検索範囲	説明
sub	クエリーに指定されるベース DN のサブツリー全体を考慮します。

表14.2 LDAP 逆参照動作

逆参照動作	説明
never	LDAP ツリーにあるエイリアスを逆参照しません。
search	検索中に見つかったエイリアスのみを逆参照します。
base	ベースオブジェクトを検索中にエイリアスのみを逆参照します。
always	LDAP ツリーにあるすべてのエイリアスを常に逆参照します。

ユーザー定義の名前マッピング

ユーザー定義の名前マッピングは、Red Hat OpenShift Service on AWS グループの名前を、LDAP サーバーでグループを検索する一意の識別子に明示的にマップします。マッピングは通常の YAML 構文を使用します。ユーザー定義のマッピングには LDAP サーバーのすべてのグループのエントリーを含めることも、それらのグループのサブセットのみを含めることもできます。ユーザー定義の名前マッピングを持たないグループが LDAP サーバーにある場合、同期時のデフォルトの動作では、Red Hat OpenShift Service on AWS グループの名前として指定された属性が使用されます。

ユーザー定義の名前マッピング

```
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. RFC 2307 設定ファイルについて

RFC 2307 スキーマでは、ユーザーエントリーとグループエントリーの両方の LDAP クエリー定義と、内部 Red Hat OpenShift Service on AWS レコードでそれらのエントリーを表すのに使用する属性を指定する必要があります。

明確にするために、Red Hat OpenShift Service on AWS で作成するグループでは、ユーザーまたは管理者側のフィールドに識別名以外の属性を可能な限り使用する必要があります。たとえば、Red Hat OpenShift Service on AWS グループのユーザーをメールアドレスで識別し、グループの名前を一般名として使用します。以下の設定ファイルでは、このような関係を作成しています。

注記

ユーザー定義名のマッピングを使用する場合、設定ファイルは異なります。

RFC 2307 スキーマを使用する LDAP 同期設定: 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) このグループのレコードが保存される LDAP サーバーの IP アドレスとホストです。

(2) **false** の場合、セキュアな LDAP (**ldaps://**) URL は TLS を使用して接続し、非セキュアな LDAP (**ldap://**) URL は TLS にアップグレードされます。 **true** の場合、サーバーへの TLS 接続は行われません。**ldaps://** URL スキームは使用できません。

(3) LDAP サーバーのグループを一意に識別する属性です。**groupUIDAttribute** に DN を使用している場合、**groupsQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイトリスト/ブラックリストの方法を使用します。

(4) グループの名前として使用する属性。

(5) メンバーシップ情報を保存するグループの属性です。

(6) LDAP サーバーでユーザーを一意に識別する属性です。userUIDAttribute に DN を使用している場合は、**usersQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイトリスト/ブラックリストの方法を使用します。

(7) Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。

14.1.2. Active Directory 設定ファイルについて

Active Directory スキーマでは、ユーザーエントリーの LDAP クエリー定義と、内部 Red Hat OpenShift Service on AWS グループレコードでそれらのエントリーを表すのに使用する属性を指定する必要があります。

明確にするために、Red Hat OpenShift Service on AWS で作成するグループでは、ユーザーまたは管理者側のフィールドに識別名以外の属性を可能な限り使用する必要があります。たとえば、Red Hat OpenShift Service on AWS グループのユーザーをメールアドレスで識別し、グループ名を LDAP サーバーのグループ名で定義します。以下の設定ファイルでは、このような関係を作成しています。

Active Directory スキーマを使用する LDAP 同期設定: 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. Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。
2. メンバーシップ情報を保存するユーザーの属性です。

14.1.3. 拡張された Active Directory 設定ファイルについて

拡張された Active Directory スキーマでは、ユーザーエントリーとグループエントリーの両方の LDAP クエリー定義と、内部 Red Hat OpenShift Service on AWS グループレコードでそれらのエントリーを表すのに使用する属性を指定する必要があります。

明確にするために、Red Hat OpenShift Service on AWS で作成するグループでは、ユーザーまたは管理者側のフィールドに識別名以外の属性を可能な限り使用する必要があります。たとえば、Red Hat OpenShift Service on AWS グループのユーザーをメールアドレスで識別し、グループの名前を一般名として使用します。以下の設定ファイルではこのような関係を作成しています。

拡張された Active Directory スキーマを使用する LDAP 同期設定: augmented_active_directory_config.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
augmentedActiveDirectory:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
        pageSize: 0
    groupUIDAttribute: dn (1)
    groupNameAttributes: [ cn ] (2)
    usersQuery:
        baseDN: "ou=users,dc=example,dc=com"
        scope: sub
        derefAliases: never
        filter: (objectclass=person)
        pageSize: 0
    userNameAttributes: [ mail ] (3)
    groupMembershipAttributes: [ memberOf ] (4)
```

1. LDAP サーバーのグループを一意に識別する属性です。groupUIDAttribute に DN を使用している場合は **groupsQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイト
2. グループの名前として使用する属性。
3. Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。
4. メンバーシップ情報を保存するユーザーの属性です。

14.2. LDAP 同期の実行

同期設定ファイルを作成後、同期を開始できます。Red Hat OpenShift Service on AWS では、管理者は同じサーバーを使用して多数の異なる同期タイプを実行できます。

14.2.1. LDAP サーバーと Red Hat OpenShift Service on AWS の同期

LDAP サーバーのすべてのグループを Red Hat OpenShift Service on AWS と同期できます。

前提条件

- 同期設定ファイルを作成します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- LDAP サーバーのすべてのグループを Red Hat OpenShift Service on AWS と同期するには、以下を実行します。

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

注記

デフォルトでは、すべてのグループ同期操作がドライランされるため、Red Hat OpenShift Service on AWS グループレコードを変更するには、**oc adm groups sync** コマンドで **--confirm** フラグを設定する必要があります。

14.2.2. Red Hat OpenShift Service on AWS グループと LDAP サーバーの同期

設定ファイルで指定した LDAP サーバー内のグループに対応する、Red Hat OpenShift Service on AWS 内のすべてのグループを同期できます。

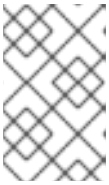
前提条件

- 同期設定ファイルを作成します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- Red Hat OpenShift Service on AWS グループを LDAP サーバーと同期するには、以下を実行します。

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



注記

デフォルトでは、すべてのグループ同期操作がドライランされるため、Red Hat OpenShift Service on AWS グループレコードを変更するには、**oc adm groups sync** コマンドで **--confirm** フラグを設定する必要があります。

14.2.3. LDAP サーバーのサブグループと Red Hat OpenShift Service on AWS の同期

LDAP グループのサブセットを、ホワイトリストファイル、ブラックリストファイル、またはその両方を使用して、Red Hat OpenShift Service on AWS と同期できます。

注記

ブラックリストファイル、ホワイトリストファイル、またはホワイトリストのリテラルの組み合わせを使用できます。ホワイトリストおよびブラックリストのファイルには 1 行ごとに 1 つの固有のグループ識別子を含める必要があり、ホワイトリストのリテラルはコマンド自体に直接含めることができます。これらのガイドラインは、Red Hat OpenShift Service on AWS にすでに存在するグループだけでなく、LDAP サーバー上にあるグループにも適用されます。

前提条件

- 同期設定ファイルを作成します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- LDAP グループのサブセットを Red Hat OpenShift Service on AWS と同期するには、次のコマンドのいずれかを使用します。

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

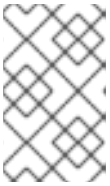
```
$ oc adm groups sync --blacklist=<blacklist_file> \
                     --sync-config=config.yaml      \
                     --confirm
```

```
$ 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
```



注記

デフォルトでは、すべてのグループ同期操作がドライランされるため、Red Hat OpenShift Service on AWS グループレコードを変更するには、**oc adm groups sync** コマンドで **--confirm** フラグを設定する必要があります。

14.3. グループのプルーニングジョブの実行

グループを作成した LDAP サーバーのレコードが存在しなくなった場合、管理者は Red Hat OpenShift Service on AWS レコードからグループを削除することもできます。プルーニングジョブは、同期ジョブに使用されるものと同じ同期設定ファイルおよびホワイトリストまたはブラックリストを受け入れます。

以下に例を示します。

```
$ 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 グループの同期の例

このセクションには、RFC 2307、Active Directory、および拡張 Active Directory スキーマについての例が記載されています。

注記

これらの例では、すべてのユーザーがそれぞれのグループの直接的なメンバーであることを想定しています。とくに、グループには他のグループがメンバーとして含まれません。ネスト化されたグループを同期する方法の詳細については、ネスト化されたメンバーシップ同期の例について参照してください。

14.4.1. RFC 2307 スキーマの使用によるグループの同期

RFC 2307 スキーマの場合、以下の例では 2 名のメンバー (**Jane** と **Jim**) を持つ **admins** というグループを同期します。以下に例を示します。

- グループとユーザーが LDAP サーバーに追加される方法。
- 同期後に生成される Red Hat OpenShift Service on AWS のグループレコード。

注記

これらの例では、すべてのユーザーがそれぞれのグループの直接的なメンバーであることを想定しています。とくに、グループには他のグループがメンバーとして含まれません。ネスト化されたグループを同期する方法の詳細については、ネスト化されたメンバーシップ同期の例について参照してください。

RFC 2307 スキーマでは、ユーザー (Jane と Jim) とグループの両方がファーストクラスエントリーとして LDAP サーバーに存在し、グループメンバーシップはグループの属性に保存されます。以下の **ldif** のスニペットでは、このスキーマのユーザーとグループを定義しています。

RFC 2307 スキーマを使用する LDAP エントリー: 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. このグループは LDAP サーバーのファーストクラスエントリーです。
2. グループのメンバーは、グループの属性としての識別参照と共にリスト表示されます。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **rfc2307_config.yaml** ファイルと同期します。

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

Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

rfc2307_config.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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) この Red Hat OpenShift Service on AWS グループが LDAP サーバーと最後に同期された時刻 (ISO 6801 形式)。

(2) LDAP サーバーのグループの固有識別子です。

(3) このグループのレコードが保存される LDAP サーバーの IP アドレスとホストです。

(4) 同期ファイルが指定するグループ名です。

(5) グループのメンバーのユーザーです。同期ファイルで指定される名前が使用されます。

14.4.2. ユーザー定義の名前マッピングに関する RFC2307 スキーマを使用したグループの同期

グループとユーザー定義の名前マッピングを同期する場合、設定ファイルは、以下に示すこれらのマッピングが含まれるように変更されます。

ユーザー定義の名前マッピングに関する RFC 2307 スキーマを使用する LDAP 同期設定: 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) ユーザー定義の名前マッピングです。

(2) ユーザー定義の名前マッピングでキーに使用される固有の識別属性です。groupUIDAttribute に DN を使用している場合は **groupsQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイトリスト/ブラックリストの方法を使用します。

(3) Red Hat OpenShift Service on AWS グループの固有の識別子がユーザー定義の名前マッピングにない場合に、Red Hat OpenShift Service on AWS グループに名前を付けるための属性。

(4) LDAP サーバーでユーザーを一意に識別する属性です。userUIDAttribute に DN を使用している場合は、**usersQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイトリスト/ブラックリストの方法を使用します。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **rfc2307_config_user_defined.yaml** ファイルとの同期を実行します。

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

Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

rfc2307_config_user_define.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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 ユーザー定義の名前マッピングが指定するグループ名です。

14.4.3. ユーザー定義のエラートレランスに関する RFC 2307 の使用によるグループの同期

デフォルトでは、同期されるグループにメンバークエリーで定義された範囲外にあるエントリーを持つメンバーが含まれる場合、グループ同期は以下のエラーを出して失敗します。

```
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>")".
```

これは **usersQuery** フィールドの **baseDN** の設定が間違っていることを示していることがよくあります。ただし、**baseDN** にグループの一部のメンバーが意図的に含まれていない場合、**tolerateMemberOutOfScopeErrors: true** を設定することでグループ同期が継続されます。範囲外のメンバーは無視されます。

同様に、グループ同期プロセスでグループのメンバーの検出に失敗した場合、同期はエラーを出して失敗します。

```
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".
```

これは **usersQuery** フィールドの設定が間違っていることを示していることがよくあります。ただし、グループに欠落していると認識されているメンバーエントリーが含まれる場合、**tolerateMemberNotFoundErrors: true** を設定することでグループ同期が継続されます。問題のあるメンバーは無視されます。

警告

LDAP グループ同期のエラートレランスを有効にすると、同期プロセスは問題のあるメンバーエントリーを無視します。LDAP グループ同期が正しく設定されていない場合、同期された Red Hat OpenShift Service on AWS グループのメンバーが欠落する可能性があります。

問題のあるグループメンバーシップに関する RFC 2307 スキーマを使用する LDAP エントリー: 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 サーバーに存在しないメンバーです。
2. 存在する可能性はあるが、同期ジョブのユーザークエリーでは **baseDN** に存在しないメンバーです。

上記の例でエラーを許容するには、以下を同期設定ファイルに追加する必要があります。

エラーを許容する RFC 2307 スキーマを使用した LDAP 同期設定: rfc2307_config_tolerating.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
rfc2307:
    groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"
        scope: sub
        derefAliases: never
    groupUIDAttribute: dn
    groupNameAttributes: [ cn ]
    groupMembershipAttributes: [ member ]
    usersQuery:
        baseDN: "ou=users,dc=example,dc=com"
        scope: sub
        derefAliases: never
```

```
userUIDAttribute: dn 1
userNameAttributes: [ mail ]
tolerateMemberNotFoundErrors: true 2
tolerateMemberOutOfScopeErrors: true 3
```

1 LDAP サーバーでユーザーを一意に識別する属性です。userUIDAttribute に DN を使用している場合は、**usersQuery** フィルターを指定できません。詳細なフィルターを実行するには、ホワイトリスト/ブラックリストの方法を使用します。

2 **true** の場合、同期ジョブは一部のメンバーが見つからなかったグループを許容し、LDAP エントリーが見つからなかったメンバーは無視されます。グループのメンバーが見つからない場合、同期ジョブのデフォルト動作は失敗します。

3 **true** の場合、同期ジョブは、一部のメンバーが **usersQuery** ベース DN で指定されるユーザー範囲外にいるグループを許容し、メンバークエリー範囲外のメンバーは無視されます。グループのメンバーが範囲外の場合、同期ジョブのデフォルト動作は失敗します。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **rfc2307_config_tolerating.yaml** ファイルを使用して同期を実行します。

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

Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

rfc2307_config.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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 同期ファイルで指定されるグループのメンバーのユーザーです。検索中に許容されるエラーがないメンバーです。

14.4.4. Active Directory スキーマの使用によるグループの同期

Active Directory スキーマでは、両方のユーザー (Jane と Jim) がファーストクラスエントリーとして LDAP サーバーに存在し、グループメンバーシップはユーザーの属性に保存されます。以下の **ldif** のスニペットでは、このスキーマのユーザーとグループを定義しています。

Active Directory スキーマを使用する LDAP エントリー: 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) ユーザーのグループメンバーシップはユーザーの属性としてリスト表示され、グループはサーバー上にエントリーとして存在しません。**memberOf** 属性はユーザーのリテラル属性である必要はありません。 一部の LDAP サーバーでは、これは検索中に作成され、クライアントに返されますが、データベースにコミットされません。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **active_directory_config.yaml** ファイルを使用して同期を実行します。

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

  Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

  active_directory_config.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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. この Red Hat OpenShift Service on AWS グループが LDAP サーバーと最後に同期された時刻 (ISO 6801 形式)。
2. LDAP サーバーのグループの固有識別子です。
3. このグループのレコードが保存される LDAP サーバーの IP アドレスとホストです。
4. LDAP サーバーにリスト表示されるグループ名です。
5. グループのメンバーのユーザーです。同期ファイルで指定される名前が使用されます。

14.4.5. 拡張された Active Directory スキーマの使用によるグループの同期

拡張された Active Directory スキーマでは、両方のユーザー (Jane と Jim) とグループがファーストクラスエントリーとして LDAP サーバーに存在し、グループメンバーシップはユーザーの属性に保存されます。以下の **ldif** のスニペットでは、このスキーマのユーザーとグループを定義しています。

拡張された Active Directory スキーマを使用する LDAP エントリー: augmented_active_directory.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com (1)

dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
```

```
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

dn: cn=admins,ou=groups,dc=example,dc=com (2)
objectClass: groupOfNames
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com
member: cn=Jim,ou=users,dc=example,dc=com
```

1. ユーザーのグループメンバーシップはユーザーの属性としてリスト表示されます。
2. このグループは LDAP サーバーのファーストクラスエントリーです。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **augmented_active_directory_config.yaml** ファイルを使用して同期を実行します。

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

Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

augmented_active_directory_config.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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. この Red Hat OpenShift Service on AWS グループが LDAP サーバーと最後に同期された時刻 (ISO 6801 形式)。
2. LDAP サーバーのグループの固有識別子です。
3. このグループのレコードが保存される LDAP サーバーの IP アドレスとホストです。
4. 同期ファイルが指定するグループ名です。
5. グループのメンバーのユーザーです。同期ファイルで指定される名前が使用されます。

14.4.5.1. LDAP のネスト化されたメンバーシップ同期の例

Red Hat OpenShift Service on AWS のグループはネストされません。LDAP サーバーはデータが使用される前にグループメンバーシップを平坦化する必要があります。Microsoft の Active Directory Server は、**LDAP_MATCHING_RULE_IN_CHAIN** ルールによりこの機能をサポートしており、これには OID **1.2.840.113556.1.4.1941** が設定されています。さらに、このマッチングルールを使用すると、明示的にホワイトリスト化されたグループのみを同期できます。

このセクションでは、拡張された Active Directory スキーマの例を取り上げ、1 名のユーザー **Jane** と 1 つのグループ **otheradmins** をメンバーとして持つ **admins** というグループを同期します。**otheradmins** グループには 1 名のユーザーメンバー **Jim** が含まれます。この例では以下のことを説明しています。

- グループとユーザーが LDAP サーバーに追加される方法。
- LDAP 同期設定ファイルの概観。
- 同期後に生成される Red Hat OpenShift Service on AWS のグループレコード。

拡張された Active Directory スキーマでは、ユーザー (**Jane** と **Jim**) とグループの両方がファーストクラスエントリーとして LDAP サーバーに存在し、グループメンバーシップはユーザーまたはグループの属性に保存されます。以下の **ldif** のスニペットはこのスキーマのユーザーとグループを定義します。

ネスト化されたメンバーを持つ拡張された Active Directory スキーマを使用する LDAP エントリー: augmented_active_directory_nested.ldif

```
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

dn: cn=Jane,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jane
sn: Smith
displayName: Jane Smith
mail: jane.smith@example.com
memberOf: cn=admins,ou=groups,dc=example,dc=com (1)

dn: cn=Jim,ou=users,dc=example,dc=com
objectClass: person
objectClass: organizationalPerson
```

```
objectClass: inetOrgPerson
objectClass: testPerson
cn: Jim
sn: Adams
displayName: Jim Adams
mail: jim.adams@example.com
memberOf: cn=otheradmins,ou=groups,dc=example,dc=com 2

dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

dn: cn=admins,ou=groups,dc=example,dc=com 3
objectClass: group
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com
member: cn=otheradmins,ou=groups,dc=example,dc=com

dn: cn=otheradmins,ou=groups,dc=example,dc=com 4
objectClass: group
cn: otheradmins
owner: cn=admin,dc=example,dc=com
description: Other System Administrators
memberOf: cn=admins,ou=groups,dc=example,dc=com 5 6
member: cn=Jim,ou=users,dc=example,dc=com
```

1 2 5 ユーザーとグループのメンバーシップはオブジェクトの属性としてリスト表示されます。

3 4 このグループは LDAP サーバーのファーストクラスエントリーです。

6 **otheradmins** グループは **admins** グループのメンバーです。

ネストされたグループを Active Directory と同期する場合、ユーザーエントリーとグループエントリーの両方の LDAP クエリー定義と、内部 Red Hat OpenShift Service on AWS グループレコードでそれらのエントリーを表すのに使用する属性を指定する必要があります。さらに、この設定では特定の変更が必要となります。

- **oc adm groups sync** コマンドはグループを明示的にホワイトリスト化する必要があります。
- ユーザーの **groupMembershipAttributes** には **"memberOf:1.2.840.113556.1.4.1941:"** を含め、**LDAP_MATCHING_RULE_IN_CHAIN** ルールに従う必要があります。
- **groupUIDAttribute** は **dn** に設定される必要があります。
- **groupsQuery**:
  - **filter** を設定しないでください。
  - 有効な **derefAliases** を設定する必要があります。
  - **baseDN** を設定しないでください。 この値は無視されます。
  - **scope** を設定しないでください。 この値は無視されます。

明確にするために、Red Hat OpenShift Service on AWS で作成するグループでは、ユーザーまたは管理者側のフィールドに識別名以外の属性を可能な限り使用する必要があります。たとえば、Red Hat OpenShift Service on AWS グループのユーザーをメールアドレスで識別し、グループの名前を一般名として使用します。以下の設定ファイルでは、このような関係を作成しています。

ネスト化されたメンバーを持つ拡張された Active Directory スキーマを使用する LDAP 同期設定です。augmented_active_directory_config_nested.yaml

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://LDAP_SERVICE_IP:389
augmentedActiveDirectory:
  groupsQuery: 1
    derefAliases: never
    pageSize: 0
  groupUIDAttribute: dn 2
  groupNameAttributes: [ cn ] 3
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=person)
    pageSize: 0
  userNameAttributes: [ mail ] 4
  groupMembershipAttributes: [ "memberOf:1.2.840.113556.1.4.1941:" ] 5
```

1. **groupsQuery** フィルターは指定できません。**groupsQuery** ベース DN およびスコープの値は無視されます。**groupsQuery** では有効な **derefAliases** を設定する必要があります。
2. LDAP サーバーのグループを一意に識別する属性です。**dn** に設定される必要があります。
3. グループの名前として使用する属性。
4. Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。ほとんどのインストールでは、**mail** または **sAMAccountName** を使用することが推奨されます。
5. メンバーシップ情報を保存するユーザーの属性です。**LDAP_MATCHING_RULE_IN_CHAIN** を使用することに注意してください。

前提条件

- 設定ファイルを保存します。
- **dedicated-admin** ロールを持つユーザーとしてクラスターにアクセスできる。

手順

- **augmented_active_directory_config_nested.yaml** ファイルを使用して同期を実行します。

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

注記

cn=admins,ou=groups,dc=example,dc=com グループを明示的にホワイトリスト化する必要があります。

Red Hat OpenShift Service on AWS は、上記の同期操作の結果として次のグループレコードを作成します。

augmented_active_directory_config_nested.yaml ファイルを使用して作成される Red Hat OpenShift Service on AWS グループ

```
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. この Red Hat OpenShift Service on AWS グループが LDAP サーバーと最後に同期された時刻 (ISO 6801 形式)。
2. LDAP サーバーのグループの固有識別子です。
3. このグループのレコードが保存される LDAP サーバーの IP アドレスとホストです。
4. 同期ファイルが指定するグループ名です。
5. グループのメンバーのユーザーです。同期ファイルで指定される名前が使用されます。グループメンバーシップは Microsoft Active Directory Server によって平坦化されているため、ネスト化されたグループのメンバーが含まれることに注意してください。

14.5. LDAP 同期設定の仕様

設定ファイルのオブジェクト仕様は以下で説明されています。スキーマオブジェクトにはそれぞれのフィールドがあることに注意してください。たとえば、v1.ActiveDirectoryConfig には **groupsQuery** フィールドがありませんが、v1.RFC2307Config と v1.AugmentedActiveDirectoryConfig の両方にこのフィールドがあります。

重要

バイナリー属性はサポートされていません。LDAP サーバーの全属性データは、UTF-8 エンコード文字列の形式である必要があります。たとえば、ID 属性として、バイナリー属性を使用することはできません (例: **objectGUID**)。代わりに **sAMAccountName** または **userPrincipalName** などの文字列属性を使用する必要があります。

14.5.1. v1.LDAPSyncConfig

LDAPSyncConfig は、LDAP グループ同期を定義するために必要な設定オプションを保持します。

名前	説明	スキーマ
kind	このオブジェクトが表す REST リソースを表す文字列の値です。サーバーはクライアントが要求を送信するエンドポイントからこれを推測できることがあります。更新はできません。CamelCase。詳細については、https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#types-kinds を参照してください。	文字列
apiVersion	オブジェクトのこの表現のバージョンスキーマを定義します。サーバーは認識されたスキーマを最新の内部値に変換し、認識されない値は拒否することがあります。詳細については、https://github.com/kubernetes/community/blob/master/contributors/devel/sig-architecture/api-conventions.md#resources を参照してください。	文字列
url	ホストは接続先の LDAP サーバーのスキーム、ホストおよびポートになります。**scheme://host:port**	文字列
bindDN	LDAP サーバーをバインドする任意の DN です。	文字列
bindPassword	検索フェーズでバインドする任意のパスワードです。	v1.StringSource

名前	説明	スキーマ
insecure	**true** の場合、接続に TLS を使用できないことを示唆します。**false** の場合、**ldaps://** URL は TLS を使用して接続し、**ldap://** URL は、https://tools.ietf.org/html/rfc2830 で指定されるように StartTLS を使用して TLS 接続にアップグレードされます。**insecure** を **true** に設定すると、**ldaps://** URL スキームを使用することはできません。	ブール値
ca	サーバーへ要求を行う際に使用する任意の信頼された認証局バンドルです。空の場合、デフォルトのシステムルートが使用されます。	文字列
groupUIDNameMapping	LDAP グループ UID から Red Hat OpenShift Service on AWS グループ名への直接マッピングです (省略可能)。	オブジェクト
rfc2307	RFC2307 と同じ方法でセットアップされた LDAP サーバーからデータを抽出するための設定を保持します。ファーストクラスグループとユーザーエントリーを抽出し、グループメンバーシップはメンバーをリスト表示するグループエントリーの複数値の属性によって決定されます。	v1.RFC2307Config
activeDirectory	Active Directory に使用されるのと同じ方法でセットアップされた LDAP サーバーからデータを抽出するための設定を保持します。ファーストクラスユーザーエントリーを抽出し、グループメンバーシップはメンバーが属するグループをリスト表示するメンバーの複数値の属性によって決定されます。	v1.ActiveDirectoryConfig

名前	説明	スキーマ
augmentedActiveDirectory	上記の Active Directory で使用されるのと同じ方法でセットアップされた LDAP サーバーからデータを抽出するための設定を保持します。1 つの追加として、ファーストクラスグループエントリーが存在し、それらはメタデータを保持するために使用されますが、グループメンバーシップは設定されません。	v1.AugmentedActiveDirectoryConfig

14.5.2. v1.StringSource

StringSource によって文字列インラインを指定できます。または環境変数またはファイルを使用して外部から指定することもできます。 文字列の値のみを含む場合、単純な JSON 文字列にマーシャルします。

名前	説明	スキーマ
value	クリアテキスト値、または **keyFile** が指定されている場合は暗号化された値を指定します。	文字列
env	クリアテキスト値、または **keyFile** が指定されている場合は暗号化された値を含む環境変数を指定します。	文字列
file	クリアテキスト値、または **keyFile** が指定されている場合は暗号化された値を含むファイルを参照します。	文字列
keyFile	値を復号化するために使用するキーを含むファイルを参照します。	文字列

14.5.3. v1.LDAPQuery

LDAPQuery は LDAP クエリーの作成に必要なオプションを保持します。

名前	説明	スキーマ
baseDN	すべての検索が開始されるディレクトリーのブランチの DN です。	文字列

名前	説明	スキーマ
scope	検索の任意の範囲です。**base** (ベースオブジェクトのみ)、**one** (ベースレベルのすべてのオブジェクト)、**sub** (サブツリー全体) のいずれかになります。設定されていない場合は、デフォルトで **sub** になります。	文字列
derefAliases	エイリアスに関する検索の任意の動作です。**never** (エイリアスを逆参照しない)、**search** (検索中の逆参照のみ)、**base** (ベースオブジェクト検索時の逆参照のみ)、**always** (常に逆参照を行う) のいずれかになります。設定されていない場合、デフォルトで **always** になります。	文字列
timeout	応答の待機を中止するまでにサーバーへの要求を未処理のままにする時間制限 (秒単位) を保持します。これが **0** の場合、クライアント側の制限が設定されないことになります。	整数
filter	ベース DN を持つ LDAP サーバーから関連するすべてのエントリーを取得する有効な LDAP 検索フィルターです。	文字列
pageSize	LDAP エントリーで測定される、推奨される最大ページサイズです。ページサイズ **0** はページングが実行されないことを意味します。	整数

14.5.4. v1.RFC2307Config

RFC2307Config は、RFC2307 スキーマを使用してどのように LDAP グループ同期が LDAP サーバーに相互作用するかを定義するために必要な設定オプションを保持します。

名前	説明	スキーマ
groupsQuery	グループエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery

名前	説明	スキーマ
groupUIDAttribute	LDAP グループエントリーのどの属性を固有の識別子として解釈するかを定義します。(**ldapGroupUID**)	文字列
groupNameAttributes	LDAP グループエントリーのどの属性を Red Hat OpenShift Service on AWS グループに使用する名前として解釈するかを定義します。	文字列配列
groupMembershipAttributes	LDAP グループエントリーのどの属性をメンバーとして解釈するかを定義します。それらの属性に含まれる値は **UserUIDAttribute** でクエリーできる必要があります。	文字列配列
usersQuery	ユーザーエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery
userUIDAttribute	LDAP ユーザーエントリーのどの属性を固有の識別子として解釈するかを定義します。**GroupMembershipAttributes** で検出される値に対応している必要があります。	文字列
userNameAttributes	LDAP ユーザーエントリーのどの属性を Red Hat OpenShift Service on AWS ユーザー名として順番に使用するかを定義します。空でない値を持つ最初の属性が使用されます。これは **LDAPPasswordIdentityProvider** の **PreferredUsername** 設定と一致している必要があります。Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。ほとんどのインストールでは、**mail** または **sAMAccountName** を使用することが推奨されます。	文字列配列

名前	説明	スキーマ
tolerateMemberNotFoundErrors	ユーザーエントリーがない場合の LDAP 同期ジョブの動作を決定します。**true** の場合、何も検出しない ユーザーの LDAP クエリーは許容され、エラーのみがログに記録されます。**false** の場合、ユーザーのクエリーが何も検出しないと、LDAP 同期ジョブは失敗します。デフォルト値は **false** です。このフラグを **true** に設定した LDAP 同期ジョブの設定が間違っていると、グループメンバーシップが削除されることがあるため、注意してこのフラグを使用してください。	ブール値
tolerateMemberOutOfScopeErrors	範囲外のユーザーエントリーが検出される場合の LDAP 同期ジョブの動作を決定します。**true** の場合、すべてのユーザークエリーに指定されるベース DN 外のユーザーの LDAP クエリーは許容され、エラーのみがログに記録されます。**false** の場合、ユーザークエリーですべてのユーザークエリーで指定されるベース DN 外を検索すると LDAP 同期ジョブは失敗します。このフラグを **true** に設定した LDAP 同期ジョブの設定が間違っていると、ユーザーのいないグループが発生することがあるため、注意してこのフラグを使用してください。	ブール値

14.5.5. v1.ActiveDirectoryConfig

ActiveDirectoryConfig は必要な設定オプションを保持し、どのように LDAP グループ同期が Active Directory スキーマを使用して LDAP サーバーと相互作用するかを定義します。

名前	説明	スキーマ
usersQuery	ユーザーエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery

名前	説明	スキーマ
userNameAttributes	LDAP ユーザーエントリーのどの属性を Red Hat OpenShift Service on AWS ユーザー名として解釈するかを定義します。Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。ほとんどのインストールでは、**mail** または **sAMAccountName** を使用することが推奨されます。	文字列配列
groupMembershipAttributes	LDAP ユーザーのどの属性をメンバーの属するグループとして解釈するかを定義します。	文字列配列

14.5.6. v1.AugmentedActiveDirectoryConfig

AugmentedActiveDirectoryConfig は必要な設定オプションを保持し、どのように LDAP グループ同期が拡張された Active Directory スキーマを使用して LDAP サーバーに相互作用するかを定義します。

名前	説明	スキーマ
usersQuery	ユーザーエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery
userNameAttributes	LDAP ユーザーエントリーのどの属性を Red Hat OpenShift Service on AWS ユーザー名として解釈するかを定義します。Red Hat OpenShift Service on AWS グループレコードでユーザーの名前として使用する属性。ほとんどのインストールでは、**mail** または **sAMAccountName** を使用することが推奨されます。	文字列配列
groupMembershipAttributes	LDAP ユーザーのどの属性をメンバーの属するグループとして解釈するかを定義します。	文字列配列
groupsQuery	グループエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery

名前	説明	スキーマ
groupUIDAttribute	LDAP グループエントリーのどの属性を固有の識別子として解釈するかを定義します。(**ldapGroupUID**)	文字列
groupNameAttributes	LDAP グループエントリーのどの属性を Red Hat OpenShift Service on AWS グループに使用する名前として解釈するかを定義します。	文字列配列