OpenShift Container Platform 4.13

認証および認可

ユーザー認証およびユーザーとサービスのアクセス制御の設定

Last Updated: 2024-04-17

OpenShift Container Platform 4.13 認証および認可

ユーザー認証およびユーザーとサービスのアクセス制御の設定

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at
http://creativecommons.org/licenses/by-sa/3.0/
. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本書では、OpenShift Container Platform でアイデンティティープロバイダーを定義する方法を説明します。また、ロールベースのアクセス制御を使用してクラスターのセキュリティーを保護する方法についても説明します。

目次

第1章 認証および認可の概要 5
1.1. OPENSHIFT CONTAINER PLATFORM の認証および認可に関する一般的な用語集 5
1.2. OPENSHIFT CONTAINER PLATFORM での認証について 6
1.3. OPENSHIFT CONTAINER PLATFORM での認可について 7

第2章 認証について 9
2.1. ユーザー 9
2.2. グループ 9
2.3. API 認証 10

第3章 内部 OAUTH サーバーの設定 13
3.1. OPENSHIFT CONTAINER PLATFORM OAUTH サーバー 13
3.2. OAUTH トークン要求フローおよび応答 13
3.3. 内部 OAUTH サーバーのオプション 14
3.4. 内部 OAUTH サーバーのトークン期間の設定 14
3.5. 内部 OAUTH サーバーのトークンの非アクティブタイムアウトの設定 15
3.6. 内部 OAUTH サーバー URL のカスタマイズ 17
3.7. OAUTH サーバーメタデータ 18
3.8. OAUTH API イベントのトラブルシューティング 19

第4章 OAUTH クライアントの設定 21
4.1. デフォルトの OAUTH クライアント 21
4.2. 追加の OAUTH クライアントの登録 21
4.3. OAUTH クライアントのトークンの非アクティブタイムアウトの設定 22
4.4. 関連情報 23

第5章 ユーザーが所有する OAUTH アクセストークンの管理 24
5.1. ユーザーが所有する OAUTH アクセストークンのリスト表示 24
5.2. ユーザーが所有する OAUTH アクセストークンの詳細の表示 24
5.3. ユーザーが所有する OAUTH アクセストークンの削除 25

第6章 アイデンティティープロバイダー設定について 27
6.1. OPENSHIFT CONTAINER PLATFORM のアイデンティティープロバイダーについて 27
6.2. サポートされるアイデンティティープロバイダー 27
6.3. KUBEADMIN ユーザーの削除 28
6.4. アイデンティティープロバイダーパラメーター 28
6.5. アイデンティティープロバイダー CR のサンプル 29

第7章 アイデンティティープロバイダーの設定 31
7.1. HTPASSWD アイデンティティープロバイダーの設定 31
7.2. KEYSTONE アイデンティティープロバイダーの設定 37
7.3. LDAP アイデンティティープロバイダーの設定 40
7.4. BASIC 認証アイデンティティープロバイダーの設定 45
7.5. 要求ヘッダーアイデンティティープロバイダーの設定 51
7.6. GITHUB または GITHUB ENTERPRISE アイデンティティープロバイダーの設定 60
7.7. GITLAB アイデンティティープロバイダーの設定 65
7.8. GOOGLE アイデンティティープロバイダーの設定 68
7.9. OPENID CONNECT ID プロバイダーの設定 71

第8章 RBAC の使用によるパーミッションの定義および適用 79
8.1. RBAC の概要 79
8.2. プロジェクトおよび NAMESPACE 82
8.3. デフォルトプロジェクト 83

8.4. クラスターロールおよびバインディングの表示 84
8.5. ローカルのロールバインディングの表示 91
8.6. ロールのユーザーへの追加 92
8.7. ローカルロールの作成 94
8.8. クラスターロールの作成 95
8.9. ローカルロールバインディングのコマンド 95
8.10. クラスターのロールバインディングコマンド 96
8.11. クラスター管理者の作成 97

第9章 KUBEADMIN ユーザーの削除 98
9.1. KUBEADMIN ユーザー 98
9.2. KUBEADMIN ユーザーの削除 98

第10章 サービスアカウントの概要および作成 99
10.1. サービスアカウントの概要 99
10.2. サービスアカウントの作成 99
10.3. ロールをサービスアカウントに付与する例 100

第11章 USING SERVICE ACCOUNTS IN APPLICATIONS 103
11.1. サービスアカウントの概要 103
11.2. デフォルトのサービスアカウント 103
11.3. サービスアカウントの作成 105

第12章 サービスアカウントの OAUTH クライアントとしての使用 107
12.1. OAUTH クライアントとしてのサービスアカウント 107

第13章 スコープトークン 110
13.1. トークンのスコープについて 110

第14章 バインドされたサービスアカウントトークンの使用 111
14.1. バインドされたサービスアカウントトークンについて 111
14.2. 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 111
14.3. 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 114

第15章 SECURITY CONTEXT CONSTRAINTS の管理 116
15.1. SECURITY CONTEXT CONSTRAINTS について 116
15.2. 事前に割り当てられる SECURITY CONTEXT CONSTRAINTS 値について 127
15.3. SECURITY CONTEXT CONSTRAINTS の例 128
15.4. セキュリティーコンテキスト制約の作成 130
15.5. SECURITY CONTEXT CONSTRAINTS へのロールベースのアクセス 132
15.6. SECURITY CONTEXT CONSTRAINTS コマンドのリファレンス 133
15.7. 関連情報 135

第16章 POD セキュリティーアドミッションの理解と管理 136
16.1. 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 136
16.2. POD セキュリティー標準とのセキュリティーコンテキスト制約の同期 136
16.3. POD セキュリティーアドミッションの同期制御 137
16.4. POD セキュリティーアドミッションアラート 137
16.5. 関連情報 138

第17章 SYSTEM:ADMIN ユーザーの権限の借用 139
17.1. API の権限借用 139
17.2. SYSTEM:ADMIN ユーザーの権限の借用 139
17.3. SYSTEM:ADMIN グループの権限の借用 139

第18章 LDAP グループの同期 140

18.1. LDAP 同期の設定について 140
18.2. LDAP 同期の実行 145
18.3. グループのプルーニングジョブの実行 147
18.4. LDAP グループを自動的に同期する 147
18.5. LDAP グループの同期の例 151
18.6. LDAP 同期設定の仕様 164

第19章 クラウドプロバイダーの認証情報の管理 172
19.1. CLOUD CREDENTIAL OPERATOR について 172
19.2. MINT モードの使用 179
19.3. PASSTHROUGH モードの使用 184
19.4. 手動モードの使用 190
19.5. AMAZON WEB SERVICES SECURITY TOKEN SERVICE での手動モードの使用 194
19.6. GCP ワークロード ID で手動モードを使用する 208

第1章 認証および認可の概要

1.1. OPENSHIFT CONTAINER PLATFORM 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、OpenShift Container Platform 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、OpenShift Container Platform 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 OpenShift Container Platform 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。

Cloud Credential Operator

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報をカスタムリソース定義 (CRD) として管理します。

設定マップ

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 ConfigMap 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。

手動モード

手動モードでは、ユーザーは Cloud Credential Operator (CCO) の代わりにクラウド認証情報を管理します。

mint モード

mint モードは、サポートされるプラットフォームで使用する Cloud Credential Operator (CCO) の

デフォルトおよび推奨されるベストプラクティスの設定です。このモードでは、CCO は提供される管理者レベルのクラウド認証情報を使用して、必要となる特定のパーミッションのみでクラスター内のコンポーネントの新規の認証情報を作成します。

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。

node

ノードは、OpenShift Container Platform 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

OpenShift Container Platform 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。

passthrough モード

passthrough モードでは、Cloud Credential Operator (CCO) は提供されるクラウド認証情報を、コンポーネントを要求するコンポーネントに渡します。

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 にリクエストを送信できるエンティティーです。

1.2. OPENSHIFT CONTAINER PLATFORM での認証について

OpenShift Container Platform クラスターへのアクセスを制御するために、クラスター管理者は ユーザー認証 を設定し、承認されたユーザーのみがクラスターにアクセスできます。

OpenShift Container Platform クラスターと対話するには、まずユーザーが OpenShift Container Platform API に対して認証する必要があります。Open Shift Container Platform API へのリクエストで、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、要求は認証されておらず、HTTP401 エラーを受け取ります。

管理者は、次のタスクを通じて認証を設定できます。

- ID プロバイダーの設定: OpenShift Container Platform でサポートされている ID プロバイダーを定義し、クラスターに追加できます。
- 内部 OAuth サーバーの設定: OpenShift Container Platform コントロールプレーンには、設定された ID プロバイダーからユーザーの ID を判別し、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。トークンの期間と非アクティブタイムアウトを設定し、内部 OAuth サーバーの URL をカスタマイズできます。

  注記

  ユーザーは、自分が所有する OAuth トークンを表示および管理できます。

- OAuth クライアントの登録: OpenShift Container Platform には、いくつかのデフォルトの OAuth クライアントが含まれています。追加の OAuth クライアントを登録および設定できます。

  注記

  ユーザーが OAuth トークンのリクエストを送信するときは、トークンを受信して使用するデフォルトまたはカスタムの OAuth クライアントを指定する必要があります。

- Cloud Credentials Operator を使用したクラウドプロバイダークレデンシャルの管理: クラスターコンポーネントは、クラウドプロバイダークレデンシャルを使用して、クラスター関連のタスクを実行するために必要なアクセス許可を取得します。
- システム管理者ユーザーのなりすまし: システム管理者ユーザーになりすます ことで、ユーザーにクラスター管理者権限を付与できます。

1.3. OPENSHIFT CONTAINER PLATFORM での認可について

許可には、識別されたユーザーが要求されたアクションを実行するための許可を持っているかどうかを判別することが含まれます。

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。OpenShift Container Platform で認可がどのように機能するかを理解するには、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、OpenShift Container Platform 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。

以下のタスクを通じて、OpenShift Container Platform の認可を管理できます。

- 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。
- クラスターロールの作成とユーザーまたはグループへの割り当て: OpenShift Container Platform には、デフォルトのクラスターロールのセットが含まれています。追加のクラスターロールを作成して、ユーザーまたはグループに追加できます。
- cluster-admin ユーザーの作成: デフォルトでは、クラスターには**kubeadmin**というクラスター管理者が 1 人だけいます。別のクラスター管理者を作成できます。クラスター管理者を作成する前に、ID プロバイダーが設定されていることを確認してください。

  注記

  クラスター管理者ユーザーを作成したら、既存の kubeadmin ユーザーを削除して、クラスターのセキュリティーを向上させます。

- 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 OpenShift Container Platform 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 か所で管理できます。

第2章 認証について

ユーザーが OpenShift Container Platform と対話できるようにするには、まずクラスターに対して認証する必要があります。認証層は、OpenShift Container Platform API への要求に関連付けられたユーザーを識別します。その後、認可層は要求側ユーザーの情報を使用して、要求が許可されるかどうかを決定します。

管理者は、OpenShift Container Platform の認証を設定できます。

2.1. ユーザー

OpenShift Container Platform の **ユーザー** は、OpenShift Container Platform API に要求できるエンティティーです。OpenShift Container Platform **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パーミッションを付与できるアクターを表します。通常、これは OpenShift Container Platform と対話してい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 OpenShift Container Platform ユーザーが表示される方法です。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、OpenShift Container Platform 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

OpenShift Container Platform 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 OpenShift Container Platform 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. OpenShift Container Platform OAuth サーバー

OpenShift Container Platform マスターには、組み込まれた OAuth サーバーが含まれます。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。以下の OAuth クライアントは、OpenShift Container Platform API の起動時に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、認証プロキシーをこのエンドポイントの前に配置するか、OpenShift Container Platform を、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、OpenShift Container Platform 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、OpenShift Container Platform 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。

2.3.1.2. API の権限借用

OpenShift Container Platform API への要求を、別のユーザーから発信されているかのように設定できます。詳細は、Kubernetes ドキュメントの User impersonation を参照してください。

2.3.1.3. Prometheus の認証メトリック

OpenShift Container Platform は認証の試行中に以下の Prometheus システムメトリックをキャプチャーします。

- **openshift_auth_basic_password_count** は **oc login** ユーザー名およびパスワードの試行回数をカウントします。
- **openshift_auth_basic_password_count_result** は、**oc login** ユーザー名およびパスワードの試行回数を結果 (**success** または **error**) 別にカウントします。
- **openshift_auth_form_password_count** は Web コンソールのログイン試行回数をカウントします。
- **openshift_auth_form_password_count_result** は結果 (**success** または **error**) 別に Web コンソールのログイン試行回数をカウントします。
- **openshift_auth_password_total** は **oc login** および Web コンソールのログイン試行回数をカウントします。

第3章 内部 OAUTH サーバーの設定

3.1. OPENSHIFT CONTAINER PLATFORM OAUTH サーバー

OpenShift Container Platform マスターには、組み込まれた OAuth サーバーが含まれます。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。

3.2. OAUTH トークン要求フローおよび応答

OAuth サーバーは、標準的な Authorization Code Grant (認可コードによるグラント) および Implicit Grant (暗黙的グラント) の OAuth 認可フローをサポートします。

OAuth トークンを、(**openshift-challenging-client** などの) **WWW-Authenticate チャレンジ** を要求するように設定された client_id で Implicit Grant (暗黙的グラント) フロー (**response_type=token**) を使用して要求する場合、以下が **/oauth/authorize** から送られる可能性のあるサーバー応答、およびそれらの処理方法になります。

ステータス	内容	クライアント応答
302	URL フラグメントに **access_token** パラメーターを含む **Location** ヘッダー (RFC 6749 セクション 4.2.2)	**access_token** 値を OAuth トークンとして使用します。
302	**error** クエリーパラメーターを含む **Location** ヘッダー (RFC 6749 セクション 4.1.2.1)	失敗します。 オプションで **error** (およびオプションの **error_description**) クエリー値をユーザーに表示します。
302	他の **Location** ヘッダー	これらのルールを使用してリダイレクトに従い、結果を処理します。
401	**WWW-Authenticate** ヘッダーが存在する	タイプ (**Basic**、**Negotiate** など) が認識される場合にチャレンジに応答し、これらのルールを使用して要求を再送信し、結果を処理します。
401	**WWW-Authenticate** ヘッダーがない	チャレンジの認証ができません。失敗し、応答本体を表示します (これには、OAuth トークンを取得する別の方法についてのリンクまたは詳細が含まれる可能性があります)
その他	その他	失敗し、オプションでユーザーに応答本体を提示します。

3.3. 内部 OAUTH サーバーのオプション

内部 OAuth サーバーには、いくつかの設定オプションを使用できます。

3.3.1. OAuth トークン期間のオプション

内部 OAuth サーバーは以下の 2 種類のトークンを生成します。

トークン	説明
アクセストークン	API へのアクセスを付与する永続的なトークン。
認可コード	アクセストークンの交換にのみ使われる一時的なトークン。

どちらの種類のトークンにもデフォルト期間を設定できます。必要な場合は、**OAuthClient** オブジェクト定義を使用してアクセストークンの期間をオーバーライドできます。

3.3.2. OAuth 付与オプション

OAuth サーバーが、ユーザーが以前にパーミッションを付与していないクライアントに対するトークン要求を受信する場合、OAuth サーバーが実行するアクションは OAuth クライアントの付与ストラテジーによって変わります。

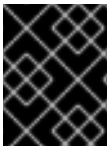
トークンを要求する OAuth クライアントは、独自の付与ストラテジーを提供する必要があります。

以下のデフォルトの方法を使用できます。

付与オプション	説明
auto	付与を自動承認し、要求を再試行します。
prompt	ユーザーに対して付与の承認または拒否を求めるプロンプトを出します。

3.4. 内部 OAUTH サーバーのトークン期間の設定

内部 OAuth サーバーのトークン期間についてのデフォルトオプションを設定できます。



重要

デフォルトで、トークンは 24 時間有効になります。24 時間を経過すると、既存のセッションは期限切れになります。

デフォルトの時間では十分ではない場合、以下の手順でこれを変更することができます。

手順

1. トークン期間オプションを含む設定ファイルを作成します。以下のファイルでは、これを、デフォルト値の 2 倍の 48 時間に設定しています。
   -

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

(1) **accessTokenMaxAgeSeconds** を設定して、アクセストークンの有効期間を制御します。デフォルトの期間は 24 時間または 86400 秒です。この属性を負の値にすることはできません。ゼロに設定すると、デフォルトの有効期間が使用されます。

2. 新規設定ファイルを適用します。

注記

既存の OAuth サーバーを更新するため、**oc apply** コマンドを使用して変更を適用する必要があります。

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

3. 変更が有効になっていることを確認します。

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

出力例

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

3.5. 内部 OAUTH サーバーのトークンの非アクティブタイムアウトの設定

OAuth トークンは、設定されるアクティブでない期間の経過後に期限切れになるように設定できます。デフォルトで、トークンの非アクティブタイムアウトは設定されません。

注記

トークンの非アクティブタイムアウトが OAuth クライアントでも設定されている場合、その値は内部 OAuth サーバー設定で設定されるタイムアウトをオーバーライドします。

前提条件

- **cluster-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。
- アイデンティティープロバイダー (IDP) を設定している。

手順

1. **OAuth** 設定を更新して、トークンの非アクティブタイムアウトを設定します。

   a. **OAuth** オブジェクトを編集します。

   ```
   $ oc edit oauth cluster
   ```

   spec.tokenConfig.accessTokenInactivityTimeout フィールドを追加し、タイムアウト値を設定します。

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

   (1) 値は適切な単位で設定します。たとえば、400 秒の場合は **400s** に、30 分の場合は **30m** に設定します。許可される最小のタイムアウト値は **300s** です。

   b. 変更を適用するためにファイルを保存します。

2. OAuth サーバー Pod が再起動していることを確認します。

   ```
   $ oc get clusteroperators authentication
   ```

   以下の出力にあるように、**PROGRESSING** が **False** と表示されるまで次の手順に移行しないでください。

   出力例

   ```
   NAME           VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
   authentication   4.13.0    True        False         False      145m
   ```

3. Kubernetes API サーバー Pod の新規リビジョンがロールアウトされていることを確認します。これには数分の時間がかかります。

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

   以下の出力にあるように、**PROGRESSING** が **False** と表示されるまで次の手順に移行しないでください。

   出力例

   ```
   NAME           VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE
   kube-apiserver   4.13.0    True        False         False      145m
   ```

   PROGRESSING が **True** と表示されている場合は、数分待機してから再試行します。

検証

1. IDP のアイデンティティーでクラスターにログインします。

2. コマンドを実行して、コマンドが正常に実行されたことを確認します。

3. アイデンティティーを使用せずに、設定されたタイムアウトよりも長く待機します。この手順の例では、400 秒よりも長い時間待機します。

4. 同じアイデンティティーのセッションからのコマンドの実行を試行します。
非アクティブの状態が設定されたタイムアウトよりも長く続くとトークンの有効期限が切れるために、このコマンドは失敗します。

出力例

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

3.6. 内部 OAUTH サーバー URL のカスタマイズ

クラスター**Ingress**設定の**spec.component Routes**フィールドでカスタムホスト名と TLS 証明書を設定することにより、内部 OAuth サーバーの URL をカスタマイズできます。

警告

内部 OAuth サーバーの URL を更新すると、OpenShift OAuth サーバーと通信して OAuth アクセストークンを取得する必要があるクラスター内のコンポーネントからの信頼が失われる可能性があります。OAuth サーバーを信頼する必要があるコンポーネントは、OAuth エンドポイントを呼び出すときに適切な CA バンドルを含める必要があります。以下に例を示します。

```
$ oc login -u <username> -p <password> --certificate-authority=<path_to_ca.crt> (1)
```

(1) 自己署名証明書の場合、**ca.crt**ファイルにカスタム CA 証明書が含まれている必要があります。含まれていない場合、ログインは成功しません。

Cluster Authentication Operator は、OAuth サーバーのサービング証明書を**openshift-config-managed** namespace の**oauth-serving-cert**設定マップに公開します。証明書は、設定マップの**data.ca-bundle.crt**キーにあります。

前提条件

- 管理者権限のあるユーザーでクラスターにログインしている。
- **openshift-config** namespace に TLS 証明書およびキーを含めたシークレットを作成している。これは、カスタムホスト名の接尾辞のドメインがクラスターのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

ヒント

oc create secret tls コマンドを使用して TLS シークレットを作成できます。

手順

1. クラスター **Ingress** 設定を編集します。

```
$ oc edit ingress.config.openshift.io cluster
```

2. カスタムのホスト名を設定し、オプションで提供する証明書とキーを設定します。

```
apiVersion: config.openshift.io/v1
kind: Ingress
metadata:
  name: cluster
spec:
  componentRoutes:
    - name: oauth-openshift
      namespace: openshift-authentication
      hostname: <custom_hostname> (1)
      servingCertKeyPairSecret:
        name: <secret_name> (2)
```

(1) カスタムホスト名。

(2) TLS 証明書 (**tls.crt**) およびキー (**tls.key**) を含む **openshift-config** namespace のシークレットへの参照。これは、カスタムホスト名の接尾辞のドメインがクラスターのドメイン接尾辞に一致しない場合に必要です。接尾辞が一致する場合には、シークレットはオプションです。

3. 変更を適用するためにファイルを保存します。

3.7. OAUTH サーバーメタデータ

OpenShift Container Platform で実行されているアプリケーションは、ビルトイン OAuth サーバーについての情報を検出する必要がある場合があります。たとえば、それらは **<namespace_route>** のアドレスを手動の設定なしで検出する必要があります。これを支援するために、OpenShift Container Platform は IETF OAuth 2.0 Authorization Server Metadata ドラフト仕様を実装しています。

そのため、クラスター内で実行されているすべてのアプリケーションは、**https://openshift.default.svc/.well-known/oauth-authorization-server** に対して **GET** 要求を実行し、以下の情報を取得できます。

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

```
  "grant_types_supported": [ 6
    "authorization_code",
    "implicit"
  ],
  "code_challenge_methods_supported": [ 7
    "plain",
    "S256"
  ]
}
```

1. **https** スキームを使用し、クエリーまたはフラグメントコンポーネントがない認可サーバーの発行者 ID です。これは、認可サーバーについての情報が含まれる **.well-known** RFC 5785 リソースが公開される場所です。
2. 認可サーバーの認可エンドポートの URL です。RFC 6749 を参照してください。
3. 認可サーバーのトークンエンドポイントの URL です。RFC 6749 を参照してください。
4. この認可サーバーがサポートする OAuth 2.0 RFC 6749 スコープの値のリストを含む JSON 配列です。サポートされるスコープの値すべてが公開される訳ではないことに注意してください。
5. この認可サーバーがサポートする OAuth 2.0 **response_type** 値のリストを含む JSON 配列です。使用される配列の値は、RFC 7591 の OAuth 2.0 Dynamic Client Registration Protocol で定義される **response_types** パラメーターで使用されるものと同じです。
6. この認可サーバーがサポートする OAuth 2.0 grant type の値のリストが含まれる JSON 配列です。使用される配列の値は、RFC 7591 の **OAuth 2.0 Dynamic Client Registration Protocol** で定義される **grant_types** パラメーターで使用されるものと同じです。
7. この認可サーバーでサポートされる PKCE RFC 7636 コードのチャレンジメソッドのリストが含まれる JSON 配列です。コード のチャレンジメソッドの値は、RFC 7636 のセクション 4.3 で定義される **code_challenge_method** パラメーターで使用されます。有効なコードのチャレンジメソッドの値は、IANA **PKCE Code Challenge Methods** レジストリーで登録される値です。IANA OAuth パラメーター を参照してください。

3.8. OAUTH API イベントのトラブルシューティング

API サーバーは、API マスターログへの直接的なアクセスがないとデバッグが困難な **unexpected condition** のエラーメッセージを返すことがあります。このエラーの根本的な理由は意図的に非表示にされます。 認証されていないユーザーにサーバーの状態についての情報を提供することを避けるためです。

これらのエラーのサブセットは、サービスアカウントの OAuth 設定の問題に関連するものです。これらの問題は、管理者以外のユーザーが確認できるイベントでキャプチャーされます。**unexpected condition** というサーバーエラーが OAuth の実行時に発生する場合、**oc get events** を実行し、これらのイベントについて **ServiceAccount** で確認します。

以下の例では、適切な OAuth リダイレクト URI がないサービスアカウントに対して警告しています。

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

出力例

```
1m        1m        1        proxy              ServiceAccount                         Warning
```

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

oc describe sa/<service_account_name> を実行すると、指定のサービスアカウント名に関連付けられた OAuth イベントが報告されます。

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

出力例

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

以下は生じる可能性のあるイベントエラーのリストです。

リダイレクト URI アノテーションが指定されていないか、無効な URI が指定されている

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

無効なルートが指定されている

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

無効な参照タイプが指定されている

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

SA トークンがない

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

第4章 OAUTH クライアントの設定

OpenShift Container Platform では、いくつかの OAuth クライアントがデフォルトで作成されます。追加の OAuth クライアントを登録し、設定することもできます。

4.1. デフォルトの OAUTH クライアント

以下の OAuth クライアントは、OpenShift Container Platform API の起動時に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。
openshift-cli-client	認可コード付与を取得するローカル HTTP サーバーを使用して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
```

4.2. 追加の OAUTH クライアントの登録

OpenShift Container Platform クラスターの認証を管理するために追加の OAuth クライアントが必要になる場合は、これを登録することができます。

手順

- 追加の OAuth クライアントを登録するには、以下を実行します。

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

(1) OAuth クライアントの **name** は、**<namespace_route>/oauth/authorize** および **<namespace_route>/oauth/token** への要求を実行する際に **client_id** パラメーターとして使用されます。

2. **secret** は、**<namespace_route>/oauth/token** への要求の実行時に **client_secret** パラメーターとして使用されます。

3. **<namespace_route>/oauth/authorize** および **<namespace_route>/oauth/token** への要求で指定される **redirect_uri** パラメーターは、**redirectURIs** パラメーター値に一覧表示されるいずれかの URI と等しいか、これによって接頭辞が付けられている必要があります。

4. **grantMethod** は、このクライアントがトークンを要求するものの、ユーザーによってアクセスが付与されていない場合に実行するアクションを判別するために使用されます。付与を自動的に承認し、要求を再試行するには **auto** を指定し、ユーザーに対して付与の承認または付与を求めるプロンプトを出す場合には **prompt** を指定します。

4.3. OAUTH クライアントのトークンの非アクティブタイムアウトの設定

OAuth クライアントを、設定されるアクティブでない期間の経過後に OAuth トークンの期限が切れるように設定できます。デフォルトで、トークンの非アクティブタイムアウトは設定されません。

注記

トークンの非アクティブタイムアウトが内部 OAuth サーバー設定でも設定されている場合、OAuth クライアントで設定されるタイムアウトはその値をオーバーライドします。

前提条件

- **cluster-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。
- アイデンティティープロバイダー (IDP) を設定している。

手順

- **OAuthClient** クライアント設定を更新して、トークンの非アクティブタイムアウトを設定します。

  a. **OAuthClient** オブジェクトを編集します。

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

  1. **<oauth_client>** を設定する OAuth クライアントに置き換えます (例: **console**)。

  accessTokenInactivityTimeoutSeconds フィールドを追加し、タイムアウト値を設定します。

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

  1. 許可される最小のタイムアウト値 (秒単位) は **300** です。

  b. 変更を適用するためにファイルを保存します。

検証

1. IDP のアイデンティティーでクラスターにログインします。設定したばかりの OAuth クライアントを使用するようにしてください。

2. アクションを実行し、これが正常に実行されたことを確認します。

3. アイデンティティーを使用せずに、設定されたタイムアウトよりも長く待機します。この手順の例では、600 秒よりも長い時間待機します。

4. 同じアイデンティティーのセッションからアクションの実行を試みます。
非アクティブの状態が設定されたタイムアウトよりも長く続くとトークンの有効期限が切れるために、この試行は失敗します。

4.4. 関連情報

- OAuthClient [oauth.openshift.io/v1]

第5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。

5.1. 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
```

5.2. 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。

5.3. 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6章 アイデンティティープロバイダー設定について

OpenShift Container Platform マスターには、組み込まれた 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。

6.1. OPENSHIFT CONTAINER PLATFORM のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、:、および % を含む OpenShift Container Platform ユーザー名はサポートされません。

6.2. 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
htpasswd	**htpasswd** アイデンティティープロバイダーを **htpasswd** を使用して生成されたフラットファイルに対してユーザー名とパスワードを検証するように設定します。
Keystone	**keystone** アイデンティティープロバイダーを、OpenShift Container Platform クラスターを Keystone に統合し、ユーザーを内部データベースに保存するように設定された OpenStack Keystone v3 サーバーによる共有認証を有効にするように設定します。
LDAP	**ldap** アイデンティティープロバイダーを、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設定します。
Basic 認証	**basic-authentication** アイデンティティープロバイダーを、ユーザーがリモートアイデンティティープロバイダーに対して検証された認証情報を使用して OpenShift Container Platform にログインできるように設定します。Basic 認証は、汎用的なバックエンド統合メカニズムです。
要求ヘッダー	**request-header** アイデンティティープロバイダーを、**X-Remote-User** などの要求ヘッダー値から識別するように設定します。通常、これは要求ヘッダー値を設定する認証プロキシーと併用されます。
GitHub または GitHub Enterprise	**github** アイデンティティープロバイダーを、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設定します。

アイデンティティープロバイダー	説明
GitLab	**gitlab** アイデンティティープロバイダーを、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設定します。
Google	**google** アイデンティティープロバイダーを、Google の OpenID Connect 統合を使用して設定します。
OpenID Connect	**oidc** アイデンティティープロバイダーを、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設定します。

アイデンティティープロバイダーが定義された後に、RBAC を使用してパーミッションの定義および適用 を実行できます。

6.3. KUBEADMIN ユーザーの削除

アイデンティティープロバイダーを定義し、新規 **cluster-admin** ユーザーを作成した後に、クラスターのセキュリティーを強化するために **kubeadmin** を削除できます。

> **警告**
>
> 別のユーザーが **cluster-admin** になる前にこの手順を実行する場合、OpenShift Container Platform は再インストールされる必要があります。このコマンドをやり直すことはできません。

前提条件

- 1 つ以上のアイデンティティープロバイダーを設定しておく必要があります。
- **cluster-admin** ロールをユーザーに追加しておく必要があります。
- 管理者としてログインしている必要があります。

手順

- **kubeadmin** シークレットを削除します。

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

6.4. 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。

6.5. アイデンティティープロバイダー CR のサンプル

以下のカスタムリソース (CR) は、アイデンティティープロバイダーを設定するために使用するパラメーターおよびデフォルト値を示します。この例では、htpasswd アイデンティティープロバイダーを使用しています。

アイデンティティープロバイダー CR のサンプル

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

1. このプロバイダー名は、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。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. **htpasswd** を使用して生成されたファイルが含まれる既存のシークレットです。

第7章 アイデンティティープロバイダーの設定

7.1. HTPASSWD アイデンティティープロバイダーの設定

htpasswd アイデンティティープロバイダーを設定して、ユーザーが htpasswd ファイルの認証情報を使用して OpenShift Container Platform にログインできるようにします。

htpasswd ID プロバイダーを定義するには、次のタスクを実行します。

1. ユーザーおよびパスワード情報を保存するために **htpasswd** ファイルを作成 します。
2. **htpasswd** ファイルを表す シークレットを作成 します。
3. シークレットを参照する htpasswd ID プロバイダーリソースを定義 します。
4. デフォルトの OAuth 設定に リソースを適用して、ID プロバイダーを追加します。

7.1.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、:、および % を含む OpenShift Container Platform ユーザー名はサポートされません。

7.1.2. htpasswd 認証について

OpenShift Container Platform で htpasswd 認証を使用すると、htpasswd ファイルに基づいてユーザーを識別できます。htpasswd ファイルは、各ユーザーのユーザー名とハッシュ化されたパスワードを含むフラットファイルです。**htpasswd** ユーティリティーを使用して、このファイルを作成できます。

7.1.3. htpasswd ファイルの作成

htpasswd ファイルの作成方法は、次のいずれかのセクションを参照してください。

- Linux を使用した htpasswd ファイルの作成
- Windows を使用した htpasswd ファイルの作成

7.1.3.1. Linux を使用した htpasswd ファイルの作成

htpasswd アイデンティティープロバイダーを使用するには、**htpasswd** を使用してクラスターのユーザー名およびパスワードを含むフラットファイルを生成する必要があります。

前提条件

- **htpasswd** ユーティリティーへのアクセスがあること。Red Hat Enterprise Linux では、これは **httpd-tools** パッケージをインストールして利用できます。

手順

1. ユーザー名およびハッシュされたパスワードを含むフラットファイルを作成します。

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

コマンドにより、ハッシュされたバージョンのパスワードが生成されます。

以下に例を示します。

```
$ htpasswd -c -B -b users.htpasswd <username> <password>
```

出力例

```
Adding password for user user1
```

2. ファイルに対する認証情報の追加またはその更新を継続します。

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

7.1.3.2. Windows を使用した htpasswd ファイルの作成

htpasswd アイデンティティープロバイダーを使用するには、**htpasswd** を使用してクラスターのユーザー名およびパスワードを含むフラットファイルを生成する必要があります。

前提条件

- **htpasswd.exe** へのアクセスがあること。このファイルは、数多くの Apache httpd ディストリビューションの **\bin** ディレクトリーに含まれます。

手順

1. ユーザー名およびハッシュされたパスワードを含むフラットファイルを作成します。

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

コマンドにより、ハッシュされたバージョンのパスワードが生成されます。

以下に例を示します。

```
> htpasswd.exe -c -B -b users.htpasswd <username> <password>
```

出力例

```
Adding password for user user1
```

2. ファイルに対する認証情報の追加またはその更新を継続します。

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

7.1.4. htpasswd シークレットの作成

htpasswd アイデンティティープロバイダーを使用するには、htpasswd ユーザーファイルが含まれるシークレットを定義する必要があります。

前提条件

- htpasswd ファイルを作成します。

手順

- htpasswd ユーザーファイルが含まれる **Secret** オブジェクトを作成します。

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

(1) 上記のコマンドが示すように、**--from-file** 引数についてのユーザーファイルを含むシークレットキーには **htpasswd** という名前を指定する必要があります。

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

7.1.5. htpasswd CR のサンプル

以下のカスタムリソース (CR) は、htpasswd アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

htpasswd CR

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

(1) このプロバイダー名は、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。

2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。

3. **htpasswd** を使用して生成されたファイルが含まれる既存のシークレットです。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.1.6. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.1.7. htpasswd アイデンティティープロバイダーの更新

既存の htpasswd アイデンティティープロバイダーにユーザーを追加したり、既存の htpasswd アイデンティティープロバイダーからユーザーを削除したりできます。

前提条件

- htpasswd ユーザーファイルが含まれる **Secret** オブジェクトを作成している。この手順では、**htpass-secret** という名前であることを前提としています。
- htpasswd アイデンティティープロバイダーを設定している。この手順では、**my_htpasswd_provider** という名前であることを前提としています。
- **htpasswd** ユーティリティーへのアクセスがある。Red Hat Enterprise Linux では、これは **httpd-tools** パッケージをインストールして利用できます。
- クラスター管理者の権限がある。

手順

1. htpasswd ファイルを **htpass-secret Secret** オブジェクトから取得し、ファイルをお使いのファイルシステムに保存します。

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

2. **users.htpasswd** ファイルからユーザーを追加したり、削除したりします。
   - 新規ユーザーを追加するには、以下を実行します。

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

出力例

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

   - 既存ユーザーを削除するには、以下を実行します。

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

出力例

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

3. **htpass-secret Secret** オブジェクトを、**users.htpasswd** ファイルの更新されたユーザーに置き換えます。

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

ヒント

または、以下の YAML を適用して Operator を置き換えることもできます。

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

4. 複数のユーザーを削除した場合は、追加でユーザーごとに既存のリソースを削除する必要があります。

   a. **User** オブジェクトを削除します。

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

   出力例

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

   ユーザーを必ず削除してください。削除しない場合、ユーザーは期限切れでない限り、トークンを引き続き使用できます。

   b. ユーザーの **Identity** オブジェクトを削除します。

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

   出力例

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

7.1.8. Web コンソールを使用したアイデンティティープロバイダーの設定

CLI ではなく Web 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 (IDP) を設定します。

前提条件

- クラスター管理者として Web コンソールにログインしている必要があります。

手順

1. **Administration → Cluster Settings** に移動します。
2. **Configuration** タブで、**OAuth** をクリックします。
3. **Identity Providers** セクションで、**Add** ドロップダウンメニューからアイデンティティープロバイダーを選択します。

注記

既存の IDP を上書きすることなく、Web コンソールで複数の IDP を指定することができます。

7.2. KEYSTONE アイデンティティープロバイダーの設定

keystone アイデンティティープロバイダーを、OpenShift Container Platform クラスターを Keystone に統合し、ユーザーを内部データベースに保存するように設定された OpenStack Keystone v3 サーバーによる共有認証を有効にするように設定します。この設定により、ユーザーは Keystone 認証情報を使用して OpenShift Container Platform にログインできます。

7.2.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.2.2. Keystone 認証について

Keystone は、アイデンティティー、トークン、カタログ、およびポリシーサービスを提供する OpenStack プロジェクトです。

新規 OpenShift Container Platform ユーザーが Keystone ユーザー名または一意の Keystone ID をベースに設定されるように Keystone との統合を設定できます。どちらの方法でも、ユーザーは Keystone ユーザー名およびパスワードを入力してログインします。OpenShift Container Platform ユーザーを Keystone ID に基づいて作成すると、より安全になります。これは、Keystone ユーザーを削除し、そのユーザー名で新しい Keystone ユーザーを作成すると、新しいユーザーが古いユーザーのリソースにアクセスできる可能性があるためです。

7.2.3. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、キーおよび証明書が含まれる **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

7.2.4. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.2.5. Keystone CR のサンプル

以下のカスタムリソース (CR) は、Keystone アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

Keystone CR

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

```
identityProviders:
- name: keystoneidp (1)
  mappingMethod: claim (2)
  type: Keystone
  keystone:
    domainName: default (3)
    url: https://keystone.example.com:5000 (4)
    ca: (5)
      name: ca-config-map
    tlsClientCert: (6)
      name: client-cert-secret
    tlsClientKey: (7)
      name: client-key-secret
```

1. このプロバイダー名は、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。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. Keystone のドメイン名です。Keystone では、ユーザー名はドメインに固有の名前です。単一ドメインのみがサポートされます。
4. Keystone サーバーへの接続に使用する URL です (必須) 。https を使用する必要があります。
5. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。
6. オプション: 設定済み URL への要求を実行する際に存在させるクライアント証明書を含む OpenShift Container Platform **Secret** オブジェクトへの参照。
7. クライアント証明書のキーを含む OpenShift Container Platform **Secret** オブジェクトへの参照。**tlsClientCert** が指定されている場合には必須になります。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.2.6. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

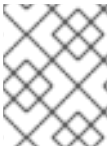
```
$ oc whoami
```

7.3. LDAP アイデンティティープロバイダーの設定

ldap アイデンティティープロバイダーを、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設定します。

7.3.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。



注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.3.2. LDAP 認証について

認証時に、指定されたユーザー名に一致するエントリーが LDAP ディレクトリーで検索されます。単一の一意の一致が見つかった場合、エントリーの識別名 (DN) と指定されたパスワードを使用した単純なバインドが試みられます。

以下の手順が実行されます。

1. 設定された **url** の属性およびフィルターとユーザーが指定したユーザー名を組み合わせて検索フィルターを生成します。

2. 生成されたフィルターを使用してディレクトリーを検索します。検索によって 1 つもエントリーが返されない場合は、アクセスを拒否します。

3. 検索で取得したエントリーの DN とユーザー指定のパスワードを使用して LDAP サーバーへのバインドを試みます。

4. バインドが失敗した場合は、アクセスを拒否します。

5. バインドが成功した場合は、アイデンティティー、電子メールアドレス、表示名、および推奨ユーザー名として設定された属性を使用してアイデンティティーを作成します。

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
```

この URL の場合:

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>))
```

たとえば、以下の URL について見てみましょう。

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

クライアントが **bob** というユーザー名を使用して接続を試みる場合、生成される検索フィルターは **(&(enabled=true)(cn=bob))** になります。

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。

7.3.3. LDAP シークレットの作成

アイデンティティープロバイダーを使用するには、**bindPassword** が含まれる OpenShift Container Platform **Secret** オブジェクトを定義する必要があります。

手順

- **bindPassword** フィールドが含まれる **Secret** オブジェクトを作成します。

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

(1) **--from-literal** 引数についての bindPassword を含むシークレットキーは **bindPassword** として指定する必要があります。

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

7.3.4. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.3.5. LDAP CR のサンプル

以下のカスタムリソース (CR) は、LDAP アイデンティティープロバイダーのパラメーターおよび許可される値を示しています。

LDAP CR

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

1. このプロバイダー名は返される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。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。

3. アイデンティティーとして使用する属性のリストです。最初の空でない属性が使用されます。少なくとも 1 つの属性が必要です。リスト表示される属性のいずれにも値がない場合、認証は失敗しま
4. メールアドレスとして使用する属性のリストです。最初の空でない属性が使用されます。
5. 表示名として使用する属性のリストです。最初の空でない属性が使用されます。
6. このアイデンティティーのユーザーをプロビジョニングする際に推奨ユーザー名として使用する属性のリストです。最初の空でない属性が使用されます。
7. 検索フェーズでバインドするために使用するオプションの DN です。**bindPassword** が定義される場合に設定される必要があります。
8. オプション: バインドパスワードを含む OpenShift Container Platform **Secret** オブジェクトへの参照。**bindDN** が定義される場合に設定される必要があります。
9. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。**insecure** が **false** の場合にのみ使用されます。
10. **true** の場合、サーバーへの TLS 接続は行われません。**false** の場合、**ldaps://** URL は TLS を使用して接続し、**ldap://** URL は TLS にアップグレードされます。これは、 **ldaps://** URL が使用されている場合は **false** に設定される必要があります。 これらの URL は常に TLS を使用して接続を試行します。
11. 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。

注記

LDAP 統合のためのユーザーのホワイトリストを作成するには、**lookup** マッピング方法を使用します。LDAP からのログインが許可される前に、クラスター管理者は各 LDAP ユーザーの **Identity** オブジェクトと **User** オブジェクトを作成する必要があります。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.3.6. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。
   -

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.4. BASIC 認証アイデンティティープロバイダーの設定

basic-authentication アイデンティティープロバイダーを、ユーザーがリモートアイデンティティープロバイダーに対して検証された認証情報を使用して OpenShift Container Platform にログインできるように設定します。Basic 認証は、汎用的なバックエンド統合メカニズムです。

7.4.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.4.2. Basic 認証について

Basic 認証は、ユーザーがリモートのアイデンティティープロバイダーに対して検証した認証情報を使用して OpenShift Container Platform にログインすることを可能にする汎用バックエンド統合メカニズムです。

Basic 認証は汎用性があるため、このアイデンティティープロバイダー使用して詳細な認証設定を実行できます。

重要

Basic 認証では、ユーザー ID とパスワードのスヌーピングを防ぎ、中間者攻撃を回避するためにリモートサーバーへの HTTPS 接続を使用する必要があります。

Basic 認証が設定されると、ユーザーはユーザー名とパスワードを OpenShift Container Platform に送信し、サーバー間の要求を行い、認証情報を Basic 認証ヘッダーとして渡すことで、これらの認証情報をリモートサーバーに対して検証することができます。このため、ユーザーはログイン時に認証情報を

OpenShift Container Platform に送信する必要があります。

注記

これはユーザー名/パスワードログインの仕組みにのみ有効で、OpenShift Container Platform はリモート認証サーバーに対するネットワーク要求を実行できる必要があります。

ユーザー名およびパスワードは Basic 認証で保護されるリモート URL に対して検証され、JSON が返されます。

401 応答は認証の失敗を示しています。

200 以外のステータスまたは空でないエラーキーはエラーを示しています。

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

sub (サブジェクト) キーを持つ **200** ステータスは、成功を示しています。

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

(1) このサブジェクトは認証ユーザーに固有である必要があり、変更することができません。

正常な応答により、以下のような追加データがオプションで提供されることがあります。

- **name** キーを使用した表示名。以下に例を示します。

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

- **email** キーを使用したメールアドレス。以下に例を示します。

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

- **preferred_username** キーを使用した推奨ユーザー名。これは、固有の変更できないサブジェクトがデータベースキーまたは UID であり、判読可能な名前が存在する場合に便利です。これは、認証されたアイデンティティーの OpenShift Container Platform ユーザーをプロビジョニングする場合にヒントとして使われます。以下に例を示します。

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

7.4.3. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、キーおよび証明書が含まれる **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

7.4.4. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.4.5. Basic 認証 CR のサンプル

以下のカスタムリソース (CR) は、Basic 認証アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

Basic 認証 CR

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

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

1. このプロバイダー名は返される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。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. Basic 認証ヘッダーで認証情報を受け入れる URL。
4. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。
5. オプション: 設定済み URL への要求を実行する際に存在させるクライアント証明書を含む OpenShift Container Platform **Secret** オブジェクトへの参照。
6. クライアント証明書のキーを含む OpenShift Container Platform **Secret** オブジェクトへの参照。**tlsClientCert** が指定されている場合には必須になります。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.4.6. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.4.7. 基本的なアイデンティティープロバイダーの Apache HTTPD 設定の例

OpenShift Container Platform 4 の基本的なアイデンティティープロバイダー (IDP) 設定では、IDP サーバーが成功および失敗について JSON で応答する必要があります。これを可能にするために、Apache HTTPD で CGI スクリプトを使用できます。本セクションでは、いくつかの例を紹介します。

例: /etc/httpd/conf.d/login.conf

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

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

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

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

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

例: /var/www/cgi-bin/login.cgi

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

例: /var/www/cgi-bin/fail.cgi

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

7.4.7.1. ファイルの要件

Apache HTTPD Web サーバーで作成するファイルの要件は以下のとおりです。

- **login.cgi** および **fail.cgi** は実行可能 (**chmod +x**) である必要があります。
- **login.cgi** および **fail.cgi** には、SELinux が有効にされている場合、適切な SELinux コンテキストがなければなりません: **restorecon -RFv /var/www/cgi-bin**、またはコンテキストが **ls -laZ** を使用して **httpd_sys_script_exec_t** であることを確認します。
- **login.cgi** は、ユーザーが **Require and Auth** ディレクティブを使用して正常にログインできる場合にのみ実行されます。
- **fail.cgi** は、ユーザーがログインに失敗する場合に実行されます (**HTTP 401** 応答が返されます)。

7.4.8. Basic 認証のトラブルシューティング

最もよく起こる問題は、バックエンドサーバーへのネットワーク接続に関連しています。簡単なデバッグの場合は、マスターで **curl** コマンドを実行します。正常なログインをテストするには、以下のコマンド例の **<user>** と **<password>** を有効な認証情報に置き換えます。無効なログインをテストするには、それらを正しくない認証情報に置き換えます。

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

正常な応答

sub (サブジェクト) キーを持つ **200** ステータスは、成功を示しています。

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

サブジェクトは認証ユーザーに固有である必要があり、変更することはできません。

正常な応答により、以下のような追加データがオプションで提供されることがあります。

- **name** キーを使用した表示名:

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

- **email** キーを使用したメールアドレス:

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

- **preferred_username** キーを使用した推奨ユーザー名:

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

  preferred_username キーは、固有の変更できないサブジェクトがデータベースキーまたは UID であり、判読可能な名前が存在する場合に便利です。これは、認証されたアイデンティティーの OpenShift Container Platform ユーザーをプロビジョニングする場合にヒントとして使われます。

失敗の応答

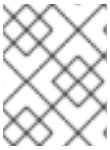
- **401** 応答は認証の失敗を示しています。
- **200** 以外のステータスまたは空でないエラーキーはエラーを示しています: **{"error":"Error message"}**

7.5. 要求ヘッダーアイデンティティープロバイダーの設定

request-header アイデンティティープロバイダーを、**X-Remote-User** などの要求ヘッダー値から識別するように設定します。通常、これは要求ヘッダー値を設定する認証プロキシーと併用されます。

7.5.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。



注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.5.2. 要求ヘッダー認証について

要求ヘッダーアイデンティティープロバイダーは、**X-Remote-User** などの要求ヘッダー値からユーザーを識別します。通常、これは要求ヘッダー値を設定する認証プロキシーと併用されます。要求ヘッダーのアイデンティティープロバイダーは、htpasswd、Keystone、LDAP、Basic 認証などの直接パスワードログインを使用する他のアイデンティティープロバイダーと組み合わせることはできません。

注記

さらに、コミュニティーでサポートされる SAML 認証 などの詳細な設定に要求ヘッダーアイデンティティープロバイダーを使用できます。このソリューションは Red Hat ではサポートされていないことに注意してください。

ユーザーがこのアイデンティティープロバイダーを使用して認証を行うには、認証プロキシー経由で **https://<namespace_route>/oauth/authorize** (およびサブパス) にアクセスする必要があります。これを実行するには、OAuth トークンに対する非認証の要求を

https://<namespace_route>/oauth/authorize にプロキシー処理するプロキシーエンドポイントにリダイレクトするよう OAuth サーバーを設定します。

ブラウザーベースのログインフローが想定されるクライアントからの非認証要求をリダイレクトするには、以下を実行します。

- **provider.loginURL** パラメーターをインタラクティブなクライアントを認証する認証プロキシー URL に設定してから、要求を **https://<namespace_route>/oauth/authorize** にプロキシーします。

WWW-Authenticate チャレンジが想定されるクライアントからの非認証要求をリダイレクトするには、以下を実行します。

- **provider.challengeURL** パラメーターを **WWW-Authenticate** チャレンジが予想されるクライアントを認証する認証プロキシー URL に設定し、要求を **https://<namespace_route>/oauth/authorize** にプロキシーします。

provider.challengeURL および **provider.loginURL** パラメーターには、URL のクエリー部分に以下のトークンを含めることができます。

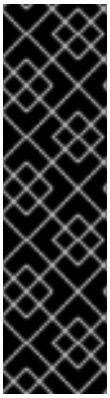
- **${url}** は現在の URL と置き換えられ、エスケープされてクエリーパラメーターで保護されます。
  例: **https://www.example.com/sso-login?then=${url}**

- **${query}** は最新のクエリー文字列と置き換えられ、エスケープされません。
  例: **https://www.example.com/auth-proxy/oauth/authorize?${query}**

重要

OpenShift Container Platform 4.1 の時点で、プロキシーは相互 TLS をサポートしている必要があります。

7.5.2.1. Microsoft Windows での SSPI 接続サポート



重要

Microsoft Windows での SSPI 接続サポートの使用はテクノロジープレビュー機能としてのみ提供されます。テクノロジープレビュー機能は、Red Hat 製品サポートのサービスレベルアグリーメント (SLA) の対象外であり、機能的に完全ではない場合があります。Red Hat は、実稼働環境でこれらを使用することを推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供して、開発段階で機能のテストを行いフィードバックを提供していただくことを目的としています。

Red Hat のテクノロジープレビュー機能のサポート範囲に関する詳細は、テクノロジープレビュー機能のサポート範囲 を参照してください。

OpenShift CLI (**oc**) は、Microsoft Windows での SSO フローを許可するために Security Support Provider Interface (SSPI) をサポートします。要求ヘッダーのアイデンティティープロバイダーを GSSAPI 対応プロキシーと共に使用して Active Directory サーバーを OpenShift Container Platform に接続する場合、ユーザーは、ドメイン参加済みの Microsoft Windows コンピューターから **oc** コマンドラインインターフェイスを使用して OpenShift Container Platform に対して自動的に認証されます。

7.5.3. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.5.4. 要求ヘッダー CR のサンプル

以下のカスタムリソース (CR) は、要求ヘッダーアイデンティティープロバイダーのパラメーターおよび許可される値を示します。

要求ヘッダー CR

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

```
nameHeaders: (9)
- X-Remote-User-Display-Name
preferredUsernameHeaders: (10)
- X-Remote-User-Login
```

1. このプロバイダー名は要求ヘッ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。

2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。

3. オプション: 非認証の **/oauth/authorize** 要求のリダイレクト先となる URL です。これは、ブラウザーベースのクライアントを認証し、その要求を **https://<namespace_route>/oauth/authorize** にプロキシーします。**https://<namespace_route>/oauth/authorize** にプロキシーする URL は **/authorize** (末尾にスラッシュはない) で終了し、OAuth 承認フローが適切に機能するようにサブパスもプロキシーする必要があります。**${url}** は現在の URL と置き換えられ、エスケープされてクエリーパラメーターで保護されます。**${query}** は最新のクエリー文字列と置き換えられます。この属性が定義されない場合は、**loginURL** が使用される必要があります。

4. オプション: 非認証の **/oauth/authorize** 要求のリダイレクト先となる URL です。これにより、**WWW-Authenticate** チャレンジが予想されるクライアントの認証が行われ、それらの要求が **https://<namespace_route>/oauth/authorize** にプロキシーされます。**${url}** は現在の URL と置き換えられ、エスケープされてクエリーパラメーターで保護されます。**${query}** は最新のクエリー文字列と置き換えられます。この属性が定義されない場合は、**challengeURL** が使用される必要があります。

5. PEM エンコードされた証明書バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。リモートサーバーによって表示される TLS 証明書を検証するためにトラストアンカーとして使用されます。

重要

OpenShift Container Platform 4.1 の時点で、**ca** フィールドはこのアイデンティティープロバイダーに必要です。これは、プロキシーが相互 TLS をサポートしている必要があることを意味します。

6. オプション: 共通名 (**cn**) のリスト。これが設定されている場合は、要求ヘッダーのユーザー名をチェックする前に指定されるリストの Common Name (**cn**) を持つ有効なクライアント証明書が提示される必要があります。空の場合、すべての Common Name が許可されます。これは **ca** と組み合わせる場合にのみ使用できます。

7. ユーザーアイデンティティーを順番にチェックする際に使用するヘッダー名。値を含む最初のヘッダーはアイデンティティーとして使用されます。これは必須であり、大文字小文字を区別します。

8. メールアドレスを順番にチェックする際に使用するヘッダー名。値を含む最初のヘッダーはメールアドレスとして使用されます。これは任意であり、大文字小文字を区別します。

9. 表示名を順番にチェックする際に使用するヘッダー名。値を含む最初のヘッダーは表示名として使用されます。これは任意であり、大文字小文字を区別します。

10. 推奨ユーザー名を順番にチェックする際に使用するヘッダー名 (**headers** に指定されるヘッダーで決定される変更不可のアイデンティティーと異なる場合)。値を含む最初のヘッダーは、プロビジョニング時に推奨ユーザー名として使用されます。これは任意であり、大文字小文字を区別します。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.5.5. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

> **注記**
>
> CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

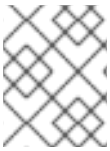
7.5.6. 要求ヘッダーを使用した Apache 認証設定の例

この例では、要求ヘッダーアイデンティティープロバイダーを使用して OpenShift Container Platform の Apache 認証プロキシーを設定します。

カスタムプロキシー設定
mod_auth_gssapi モジュールの使用は、要求ヘッダーアイデンティティープロバイダーを使用して Apache 認証プロキシーを設定する一般的な方法ですが、必須の方法ではありません。以下の要件を満たすと、他のプロキシーを簡単に使用できます。

- クライアント要求の **X-Remote-User** ヘッダーをブロックして、スプーフィングを防ぎます。
- **RequestHeaderIdentityProvider** 設定でクライアント証明書の認証を適用します。

- チャレンジフローを使用してすべての認証要求についての **X-Csrf-Token** ヘッダーを設定する必要があります。
- **/oauth/authorize** エンドポイントとそのサブパスのみがプロキシーされる必要があります。バックエンドサーバーがクライアントを正しい場所に送信できるようリダイレクトは書き換える必要があります。
- **https://<namespace_route>/oauth/authorize** にプロキシーする URL は **/authorize** で終了 (末尾のスラッシュなし) する必要があります。たとえば、**https://proxy.example.com/login-proxy/authorize?...** は、**https://<namespace_route>/oauth/authorize?...** にプロキシーする必要があります。
- **https://<namespace_route>/oauth/authorize** にプロキシーされる URL のサブパスは、**https://<namespace_route>/oauth/authorize** にプロキシーする必要があります。たとえば、**https://proxy.example.com/login-proxy/authorize/approve?...** は、**https://<namespace_route>/oauth/authorize/approve?...** にプロキシーする必要があります。



注記

https://<namespace_route> アドレスは OAuth サーバーへのルートであり、**oc get route -n openshift-authentication** を実行して取得できます。

要求ヘッダーを使用した Apache 認証の設定

この例では、**mod_auth_gssapi** モジュールを使用し、要求ヘッダーアイデンティティープロバイダーを使用して Apache 認証プロキシーを設定します。

前提条件

- **mod_auth_gssapi** モジュールを Optional チャンネル から取得します。ローカルマシンに以下のパッケージをインストールする必要があります。
  - **httpd**
  - **mod_ssl**
  - **mod_session**
  - **apr-util-openssl**
  - **mod_auth_gssapi**
- 信頼されたヘッダーを送信する要求を検証するために CA を生成します。CA を含む OpenShift Container Platform **ConfigMap** オブジェクトを定義します。これは、以下を実行して行います。

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

(1) CA は、**ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

- このプロキシー用のクライアント証明書を生成します。この証明書は、x509 証明書ツールを使用して生成できます。信頼されたヘッダーを送信する要求を検証するために、生成した CA でクライアント証明書に署名する必要があります。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。

手順

このプロキシーはクライアント証明書を使用して OAuth サーバーに接続します。これは、**X-Remote-User** ヘッダーを信頼するように設定されます。

1. Apache 設定の証明書を作成します。**SSLProxyMachineCertificateFile** パラメーターの値として指定する証明書は、プロキシーをサーバーに対して認証するために使用されるプロキシーのクライアント証明書です。これは、拡張されたキーのタイプとして **TLS Web Client Authentication** を使用する必要があります。
2. Apache 設定を作成します。以下のテンプレートを使用して必要な設定および値を指定します。

重要

テンプレートを十分に確認し、その内容を環境に合うようにカスタマイズします。

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

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

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

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

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

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

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

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

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

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

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

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

      ErrorDocument 401 /login.html
```

```
  </Location>

</VirtualHost>

RequestHeader unset X-Remote-User
```

注記

https://<namespace_route> アドレスは OAuth サーバーへのルートであり、**oc get route -n openshift-authentication** を実行して取得できます。

3. カスタムリソース (CR) の **identityProviders** スタンザを更新します。

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

4. 設定を確認します。

   a. 適切なクライアント証明書およびヘッダーを指定して、トークンを要求し、プロキシーをバイパスできることを確認します。

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

   b. クライアント証明書を提供しない要求が、証明書なしでトークンを要求して失敗することを確認します。

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

   c. **challengeURL** リダイレクトがアクティブであることを確認します。

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

   以下の手順で使用する **challengeURL** リダイレクトをコピーします。

   d. このコマンドを実行して、**WWW-Authenticate** 基本チャレンジ、ネゴシエートチャレンジ、またはそれらの両方のチャレンジを含む **401** 応答を表示します。

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

e. Kerberos チケットを使用または使用せずに、OpenShift CLI (**oc**) へのログインをテストします。

i. **kinit** を使用して Kerberos チケットを生成した場合は、これを破棄します。

```
# kdestroy -c cache_name (1)
```

(1) Kerberos キャッシュの名前を指定します。

ii. Kerberos 認証情報を使用して **oc** ツールにログインします。

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

プロンプトで、Kerberos パスワードを入力します。

iii. **oc** ツールからログアウトします。

```
# oc logout
```

iv. Kerberos 認証情報を使用してチケットを取得します。

```
# kinit
```

プロンプトで、Kerberos ユーザー名およびパスワードを入力します。

v. **oc** ツールにログインできることを確認します。

```
# oc login
```

設定が正しい場合は、別の認証情報を入力せずにログインできます。

7.6. GITHUB または GITHUB ENTERPRISE アイデンティティープロバイダーの設定

github アイデンティティープロバイダーを、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設定します。OAuth は OpenShift Container Platform 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。

GitHub 統合を使用して GitHub または GitHub Enterprise のいずれかに接続できます。GitHub Enterprise 統合の場合、インスタンスの **hostname** を指定する必要があり、サーバーへの要求で使用する **ca** 証明書バンドルをオプションで指定することができます。

注記

とくに記述がない限り、以下の手順が GitHub および GitHub Enterprise の両方に適用されます。

7.6.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、:、および % を含む OpenShift Container Platform ユーザー名はサポートされません。

7.6.2. GitHub 認証について

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 OpenShift Container Platform 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 OpenShift Container Platform 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ことができます。

7.6.3. GitHub アプリケーションの登録

GitHub または GitHub Enterprise をアイデンティティープロバイダーとして使用するには、使用するアプリケーションを登録する必要があります。

手順

1. アプリケーションを GitHub で登録します。
   - GitHub の場合、Settings → Developer settings → OAuth Apps → Register a new OAuth application をクリックします。
   - GitHub Enterprise の場合は、GitHub Enterprise ホームページに移動してから **Settings → Developer settings → Register a new application** をクリックします。
2. アプリケーション名を入力します (例: **My OpenShift Install**)。
3. ホームページ URL (例: **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>**) を入力します。
4. オプション: アプリケーションの説明を入力します。
5. 認可コールバック URL を入力します。 ここで、URL の終わりにはアイデンティティープロバイダーの **name** が含まれます。

   ```
   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/github
   ```

6. **Register application** をクリックします。GitHub はクライアント ID とクライアントシークレットを提供します。これらの値は、アイデンティティープロバイダーの設定を完了するために必要です。

7.6.4. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、文字列を含む **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

- 以下のコマンドを実行して、証明書ファイルなどのファイルの内容を含む **Secret** オブジェクトを定義できます。

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

7.6.5. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

注記

この手順は、GitHub Enterprise にのみ必要です。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.6.6. GitHub CR のサンプル

以下のカスタムリソース (CR) は、GitHub アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

GitHub CR

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

1. このプロバイダー名は GitHub の数字ユーザー ID 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。これはコールバック URL を作成するためにも使用されます。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。非公開で信頼されているルート証明書で GitHub Enterprise の場合のみ使用されます。
4. 登録済みの GitHub OAuth アプリケーション のクライアント ID。アプリケーション

は、**https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>** のコールバック URL を使用して設定する必要があります。

5. GitHub で発行されるクライアントシークレットが含まれる OpenShift Container Platform **Secret** オブジェクトへの参照。

6. GitHub Enterprise の場合、**example.com** などのインスタンスのホスト名を指定する必要があります。この値は **/setup/settings** ファイルにある GitHub Enterprise **hostname** 値に一致する必要があり、ポート番号を含めることはできません。この値が設定されない場合、**teams** または **organizations** のいずれかが定義される必要があります。GitHub の場合は、このパラメーターを省略します。

7. 組織のリストです。**hostname** フィールドが設定されていないか、**mappingMethod** が **lookup** に設定されている場合は **organizations** または **teams** フィールドを設定する必要があります。これは **teams** フィールドと組み合わせて使用することはできません。

8. チームのリストです。**hostname** フィールドが設定されていないか、**mappingMethod** が **lookup** に設定されている場合は **teams** または **organizations** フィールドのいずれかを設定する必要があります。これは **organizations** フィールドと組み合わせて使用することはできません。

注記

organizations または **teams** が指定されている場合、少なくともリストのいずれかの組織のメンバーである GitHub ユーザーのみがログインできます。その組織が **clientID** で設定された GitHub OAuth アプリケーションを所有していない場合、組織の所有者はこのオプションを使用するためにサードパーティーのアクセスを付与する必要があります。これは組織の管理者が初回の GitHub ログイン時に、または GitHub の組織設定で実行できます。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.6.7. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. OAuth サーバーからトークンを取得します。
   kubeadmin ユーザーが削除されている限り、**oc login** コマンドは、トークンを取得できる Web ページにアクセスする方法についての情報を提供します。

   Web コンソールからこのページにアクセスするには、**(?) Help → Command Line Tools→ Copy Login Command**に移動します。

3. 認証するトークンを渡して、クラスターにログインします。

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

注記

このアイデンティティープロバイダーは、ユーザー名とパスワードを使用してログインすることをサポートしません。

4. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.7. 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 ID プロバイダーとして使用して、 **gitlab** ID プロバイダーを設定します。

7.7.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.7.2. GitLab 認証について

GitLab 認証を設定することによって、ユーザーは GitLab 認証情報を使用して OpenShift Container Platform にログインできます。

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。

7.7.3. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、文字列を含む **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

- 以下のコマンドを実行して、証明書ファイルなどのファイルの内容を含む **Secret** オブジェクトを定義できます。

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

7.7.4. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

注記

この手順は、GitHub Enterprise にのみ必要です。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.7.5. GitLab CR のサンプル

以下のカスタムリソース (CR) は、GitLab アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

GitLab CR

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

1. このプロバイダー名は GitLab 数字ユーザー ID 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。これはコールバック URL を作成するためにも使用されます。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. 登録済みの GitLab OAuth アプリケーション のクライアント ID です。アプリケーションは、**https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>** のコールバック URL を使用して設定する必要があります。
4. GitLab で発行されるクライアントシークレットが含まれる OpenShift Container Platform **Secret** オブジェクトへの参照。
5. GitLab プロバイダーのホスト URL です。これは **https://gitlab.com/** か、他の GitLab の自己ホストインスタンスのいずれかになります。
6. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。

認証局ファイルを含む OpenShift Container Platform **ConfigMap** オブジェクトへの参照。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.7.6. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

> **注記**
>
> CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. アイデンティティープロバイダーのユーザーとしてクラスターにログインし、プロンプトが出されたらパスワードを入力します。

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

3. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.8. GOOGLE アイデンティティープロバイダーの設定

Google OpenID Connect 統合 を使用して **google** ID プロバイダーを設定します。

7.8.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、:、および % を含む OpenShift Container Platform ユーザー名はサポートされません。

7.8.2. Google 認証について

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。

注記

Google をアイデンティティープロバイダーとして使用するには、**<namespace_route>/oauth/token/request** を使用してトークンを取得し、コマンドラインツールで使用する必要があります。

7.8.3. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、文字列を含む **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

- 以下のコマンドを実行して、証明書ファイルなどのファイルの内容を含む **Secret** オブジェクトを定義できます。

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

7.8.4. Google CR のサンプル

以下のカスタムリソース (CR) は、Google アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

Google CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: googleidp (1)
    mappingMethod: claim (2)
    type: Google
    google:
      clientID: {...} (3)
      clientSecret: (4)
        name: google-secret
      hostedDomain: "example.com" (5)
```

1. このプロバイダー名は Google の数字のユーザー ID 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。これはリダイレクト URL を作成するためにも使用されます。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. 登録済みの Google プロジェクト のクライアント ID です。プロジェクトは、 **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>** のリダイレクト URI で設定する必要があります。
4. Google で発行されるクライアントシークレットが含まれる OpenShift Container Platform **Secret** オブジェクトへの参照。
5. サインインアカウントを制限するために使用される ホスト型ドメイン です。**lookup mappingMethod** が使用される場合はオプションになります。空の場合は、すべての Google アカウントの認証が許可されます。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.8.5. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. OAuth サーバーからトークンを取得します。
kubeadmin ユーザーが削除されている限り、**oc login** コマンドは、トークンを取得できる Web ページにアクセスする方法についての情報を提供します。

   Web コンソールからこのページにアクセスするには、**(?) Help → Command Line Tools→ Copy Login Command**に移動します。

3. 認証するトークンを渡して、クラスターにログインします。

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

注記

このアイデンティティープロバイダーは、ユーザー名とパスワードを使用してログインすることをサポートしません。

4. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.9. OPENID CONNECT ID プロバイダーの設定

oidc アイデンティティープロバイダーを、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設定します。

7.9.1. OpenShift Container Platform のアイデンティティープロバイダーについて

デフォルトでは、**kubeadmin** ユーザーのみがクラスターに存在します。アイデンティティープロバイダーを指定するには、アイデンティティープロバイダーを記述し、これをクラスターに追加するカスタムリソースを作成する必要があります。

注記

/、**:**、および **%** を含む OpenShift Container Platform ユーザー名はサポートされません。

7.9.2. OpenID Connect 認証について

OpenShift Container Platform 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。

注記

ID Token および **UserInfo** の復号化はサポートされていません。

デフォルトで、**openid** の範囲が要求されます。必要な場合は、**extraScopes** フィールドで追加の範囲を指定できます。

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 **UserInfo** 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。標準のアイデンティティー要求は **sub** になります。

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。次の表に、標準クレームを示します。

要求	説明
sub	subject identifier の省略形です。 発行側のユーザーのリモートアイデンティティーです。
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。

注記

OpenID Connect ID プロバイダーがリソース所有者パスワード認証情報 (ROPC) 付与フローをサポートしていない限り、ユーザーはコマンドラインツールで使用するために **<namespace_route>/oauth/token/request** からトークンを取得する必要があります。

7.9.3. サポートされている OIDC プロバイダー

Red Hat は、OpenShift Container Platform を使用して特定の OpenID Connect (OIDC) プロバイダーをテストし、サポートします。以下の Open IDConnect (OIDC) プロバイダーは、OpenShift Container Platform でテストおよびサポートされています。次のリストにない OIDC プロバイダーの使用は OpenShift Container Platform で動作する可能性がありますが、そのプロバイダーは Red Hat によってテストされていないため、Red Hat のサポート外となります。

- Windows Server 向けの Active Directory Federation サービス

注記

現時点では、カスタムクレームが使用される場合に、Windows Server for Windows Server 向けの Active Directory Federation Services を OpenShift Container Platform で使用することはサポートされていません。

- GitLab
- Google
- Keycloak
- Microsoft Identity Platform (Azure Active Directory v2.0)

注記

現時点で、グループ名の同期が必要な場合に Microsoft identity platform を使用することは サポートされていません。

- Okta
- Ping Identity
- Red Hat Single Sign-On

7.9.4. シークレットの作成

アイデンティティープロバイダーは **openshift-config** namespace で OpenShift Container Platform **Secret** オブジェクトを使用して、クライアントシークレット、クライアント証明書およびキーをこれに組み込みます。

手順

- 以下のコマンドを使用して、文字列を含む **Secret** オブジェクトを作成します。

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

ヒント

または、以下の YAML を適用してシークレットを作成できます。

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

- 以下のコマンドを実行して、証明書ファイルなどのファイルの内容を含む **Secret** オブジェクトを定義できます。

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

7.9.5. 設定マップの作成

アイデンティティープロバイダーは、**openshift-config** namespace で OpenShift Container Platform **ConfigMap** オブジェクトを使用し、認証局バンドルをこれに組み込みます。これらは、主にアイデンティティープロバイダーで必要な証明書バンドルを組み込むために使用されます。

注記

この手順は、GitHub Enterprise にのみ必要です。

手順

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform **ConfigMap** オブジェクトを定義します。認証局は **ConfigMap** オブジェクトの **ca.crt** キーに保存する必要があります。

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

ヒント

または、以下の YAML を適用して設定マップを作成できます。

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

7.9.6. OpenID Connect CR のサンプル

以下のカスタムリソース (CR) は、OpenID Connect アイデンティティープロバイダーのパラメーターおよび許可される値を示します。

カスタム証明書バンドル、追加の範囲、追加の認可要求パラメーター、または **userInfo** URL を指定する必要がある場合、完全な OpenID Connect CR を使用します。

標準の OpenID Connect CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: oidcidp (1)
    mappingMethod: claim (2)
    type: OpenID
```

```
  openID:
    clientID: ... (3)
    clientSecret: (4)
      name: idp-secret
    claims: (5)
      preferredUsername:
      - preferred_username
      name:
      - name
      email:
      - email
      groups:
      - groups
    issuer: https://www.idp-issuer.com (6)
```

1. このプロバイダー名はアイデンティティー要求の値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。これはリダイレクト URL を作成するためにも使用されます。
2. このプロバイダーのアイデンティティーと **User** オブジェクト間にマッピングが確立される方法を制御します。
3. OpenID プロバイダーに登録されているクライアントのクライアント ID です。このクライアントは **https://oauth-openshift.apps.<cluster-name>.<cluster-domain>/oauth2callback/<idp-provider-name>** にリダイレクトすることを許可されている必要があります。
4. クライアントシークレットを含む OpenShift Container Platform **Secret** オブジェクトへの参照。
5. アイデンティティーとして使用する要求のリストです。空でない最初の要求が使用されます。
6. OpenID 仕様に記述される発行者 ID。クエリーまたはフラグメントコンポーネントのない **https** を使用する必要があります。

完全な OpenID Connect CR

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: oidcidp
    mappingMethod: claim
    type: OpenID
    openID:
      clientID: ...
      clientSecret:
        name: idp-secret
      ca: (1)
        name: ca-config-map
      extraScopes: (2)
      - email
      - profile
      extraAuthorizeParameters: (3)
        include_granted_scopes: "true"
```

```
  claims:
    preferredUsername: 4
    - preferred_username
    - email
    name: 5
    - nickname
    - given_name
    - name
    email: 6
    - custom_email_claim
    - email
    groups: 7
    - groups
  issuer: https://www.idp-issuer.com
```

1. オプション: 設定済みの URL のサーバー証明書を検証するために使用する PEM エンコードされた認証局バンドルを含む OpenShift Container Platform 設定マップへの参照。
2. オプション: 認可トークン要求時に **openid** の範囲のほかに要求する範囲の一覧です。
3. オプション: 認可トークン要求に追加する追加パラメーターのマップです。
4. このアイデンティティーのユーザーをプロビジョニングする際に推奨ユーザー名として使用される要求のリストです。空でない最初の要求が使用されます。
5. 表示名として使用する要求のリストです。空でない最初の要求が使用されます。
6. メールアドレスとして使用する要求のリストです。空でない最初の要求が使用されます。
7. ユーザーのログイン時に Open IDConnect プロバイダーから OpenShift Container Platform にグループを同期するために使用するクレームのリスト。空でない最初の要求が使用されます。

関連情報

- 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の詳細は、アイデンティティープロバイダーのパラメーター (**mappingMethod** など) について参照してください。

7.9.7. アイデンティティープロバイダーのクラスターへの追加

クラスターのインストール後に、アイデンティティープロバイダーをそのクラスターに追加し、ユーザーの認証を実行できるようにします。

前提条件

- OpenShift Container Platform クラスターを作成します。
- アイデンティティープロバイダーのカスタムリソース (CR) を作成します。
- 管理者としてログインしている必要があります。

手順

1. 定義された CR を適用します。

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

注記

CR が存在しない場合、**oc apply** は新規 CR を作成し、さらに以下の警告をトリガーする可能性があります。**Warning: oc apply should be used on resources created by either oc create --save-config or oc apply**この場合は、この警告を無視しても問題ありません。

2. OAuth サーバーからトークンを取得します。
   kubeadmin ユーザーが削除されている限り、**oc login** コマンドは、トークンを取得できる Web ページにアクセスする方法についての情報を提供します。

   Web コンソールからこのページにアクセスするには、**(?) Help → Command Line Tools→ Copy Login Command**に移動します。

3. 認証するトークンを渡して、クラスターにログインします。

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

注記

OpenID Connect アイデンティティープロバイダーが Resource Owner Password Credentials (ROPC) Grant フローをサポートする場合、ユーザー名とパスワードを使用してログインすることができます。アイデンティティープロバイダーの ROPC Grant フローを有効にする手順を実行する必要がある場合があります。

OIDC アイデンティティープロバイダーを OpenShift Container Platform で設定した後に、以下のコマンドを使用してログインできます。この場合、ユーザー名とパスワードの入力が求めるプロンプトが出されます。

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

4. ユーザーが正常にログインされていることを確認し、ユーザー名を表示します。

```
$ oc whoami
```

7.9.8. Web コンソールを使用したアイデンティティープロバイダーの設定

CLI ではなく Web 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 (IDP) を設定します。

前提条件

- クラスター管理者として Web コンソールにログインしている必要があります。

手順

1. **Administration → Cluster Settings** に移動します。
2. **Configuration** タブで、**OAuth** をクリックします。

3. **Identity Providers** セクションで、**Add** ドロップダウンメニューからアイデンティティープロバイダーを選択します。

注記

既存の IDP を上書きすることなく、Web コンソールで複数の IDP を指定することができます。

第8章 RBAC の使用によるパーミッションの定義および適用

8.1. 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。

これにより、プラットフォーム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、OpenShift Container Platform 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。

8.1.1. デフォルトのクラスターロール

OpenShift Container Platform には、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るデフォルトのクラスターロールのセットが含まれます。

重要

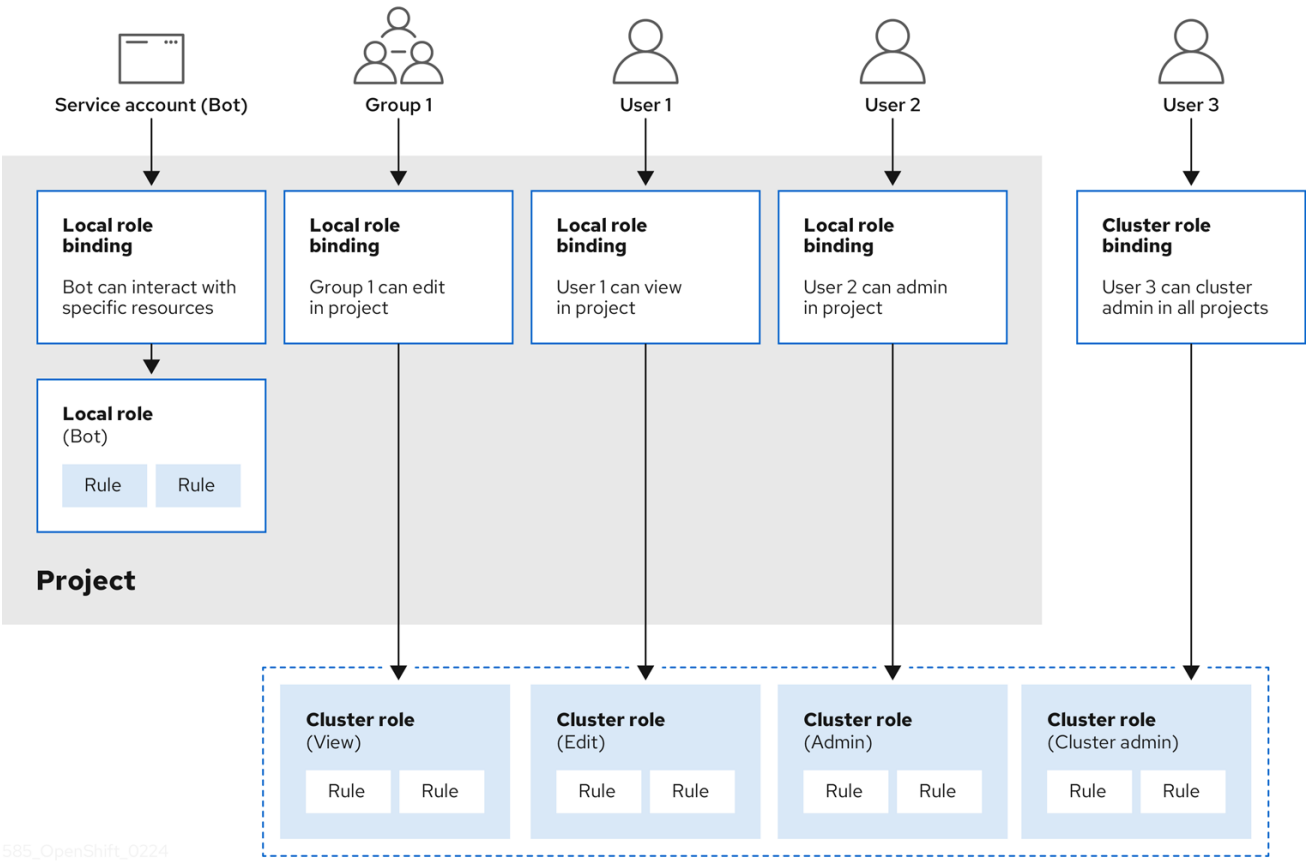
デフォルトのクラスターロールを手動で変更することは推奨されません。このようなシステムロールへの変更は、クラスターが正常に機能しなくなることがあります。

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、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。

8.1.2. 認可の評価

OpenShift Container Platform 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。

OpenShift Container Platform は以下の手順を使用し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。

8.1.2.1. 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。

8.2. 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。

クラスター管理者はプロジェクトを作成でき、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任意のメンバーに委任できます。クラスター管理者は、開発者が独自のプロジェクトを作成することも許可できます。

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。

8.3. デフォルトプロジェクト

OpenShift Container Platform 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。

注記

デフォルト namespace (**default**、**kube-system**、**kube-public**、**openshift-node**、**openshift-infra**、**openshift**) のいずれかに作成された Pod に SCC を割り当てることはできません。これらの namespace は Pod またはサービスを実行するために使用することはできません。

8.4. 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。

クラスター全体でバインドされた **cluster-admin** のデフォルトのクラスターロールを持つユーザーは、クラスターロールおよびバインディングの表示を含む、すべてのリソースでのすべてのアクションを実行できます。

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

...
```

8.5. 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。
  - クラスター全体でバインドされた **cluster-admin** のデフォルトのクラスターロールを持つユーザー は、ローカルロールおよびバインディングの表示を含む、すべてのリソースでのすべてのアクションを実行できます。
  - 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
```

8.6. 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。

8.7. 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
```

8.8. クラスターロールの作成

クラスターロールを作成できます。

手順

1. クラスター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 clusterrole <name> --verb=<verb> --resource=<resource>
```

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。

たとえば、ユーザーが Pod を閲覧できるようにするクラスター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 clusterrole podviewonly --verb=get --resource=pod
```

8.9. 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8.1 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。
$ 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。

8.10. 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8.2 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。

8.11. クラスター管理者の作成

cluster-admin ロールは、クラスターリソースの変更など、OpenShift Container Platform クラスターでの管理者レベルのタスクを実行するために必要です。

前提条件

- クラスター管理者として定義するユーザーを作成していること。

手順

- ユーザーをクラスター管理者として定義します。

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

第9章 KUBEADMIN ユーザーの削除

9.1. KUBEADMIN ユーザー

OpenShift Container Platform は、インストールプロセスの完了後にクラスター管理者 **kubeadmin** を作成します。

このユーザーには、**cluster-admin** ロールが自動的に適用され、このユーザーはクラスターの root ユーザーとしてみなされます。パスワードは動的に生成され、OpenShift Container Platform 環境に対して一意です。インストールの完了後に、パスワードはインストールプログラムの出力で提供されます。以下に例を示します。

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

9.2. KUBEADMIN ユーザーの削除

アイデンティティープロバイダーを定義し、新規 **cluster-admin** ユーザーを作成した後に、クラスターのセキュリティーを強化するために **kubeadmin** を削除できます。

> **警告**
>
> 別のユーザーが **cluster-admin** になる前にこの手順を実行する場合、OpenShift Container Platform は再インストールされる必要があります。このコマンドをやり直すことはできません。

前提条件

- 1 つ以上のアイデンティティープロバイダーを設定しておく必要があります。
- **cluster-admin** ロールをユーザーに追加しておく必要があります。
- 管理者としてログインしている必要があります。

手順

- **kubeadmin** シークレットを削除します。

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

第10章 サービスアカウントの概要および作成

10.1. 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 OpenShift Container Platform アカウントです。サービスアカウントは各プロジェクトに存在する 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。

OpenShift Container Platform 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。たとえば、サービスアカウントにより、以下が可能になります。

- レプリケーションコントローラーが Pod を作成するか、削除するために API 呼び出しを実行する。
- コンテナー内のアプリケーションが検出目的で API 呼び出しを実行する。
- 外部アプリケーションがモニターまたは統合目的で API 呼び出しを実行する。

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。

10.2. 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>
```

10.3. 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
```

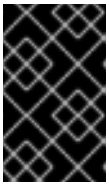
ヒント

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11章 USING SERVICE ACCOUNTS IN APPLICATIONS

11.1. 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 OpenShift Container Platform アカウントです。サービスアカウントは各プロジェクトに存在する 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。

OpenShift Container Platform 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。たとえば、サービスアカウントにより、以下が可能になります。

- レプリケーションコントローラーが Pod を作成するか、削除するために API 呼び出しを実行する。
- コンテナー内のアプリケーションが検出目的で API 呼び出しを実行する。
- 外部アプリケーションがモニターまたは統合目的で API 呼び出しを実行する。

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。

11.2. デフォルトのサービスアカウント

OpenShift Container Platform 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。

11.2.1. 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 OpenShift Container Platform インフラストラ

クチャープロジェクト (**openshift-infra**) に作成され、クラスター全体での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。
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。

11.2.2. 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。
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。 このロールは、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することを可能にします。

11.2.3. 自動生成されるサービスアカウントトークンシークレット

サービスアカウントが作成されると、そのサービスアカウント用のトークンシークレットが自動的に生成されます。このサービスアカウントトークンシークレットは、自動的に生成された Docker 設定シークレットとともに、内部 OpenShift Container Platform レジストリーに対する認証に使用されます。これらの自動生成されたシークレットは、自分での使用に依存することがないようにしてください。これらは将来の OpenShift Container Platform リリースで削除される可能性があります。

注記

OpenShift Container Platform 4.11 より前では、サービスアカウントの作成時に 2 番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。

OpenShift Container Platform 4.11 以降、この 2 番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。

4.13 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。

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。

11.3. 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12章 サービスアカウントの OAUTH クライアントとしての使用

12.1. 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。

12.1.1. 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13章 スコープトークン

13.1. 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。**cluster-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。

13.1.1. 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。

13.1.2. 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14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。

14.1. 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。

14.2. 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

- **cluster-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. オプション: サービスアカウントの発行者を設定します。
   通常、このステップはバインドされたトークンがクラスター内でのみ使用される場合には必要ありません。

   重要

   サービスアカウントの発行者をカスタムのものに変更した場合、以前のサービスアカウントの発行者は引き続き 24 時間信頼されます。

   クラスター内のすべての Pod を手動で再起動するか、ノードのローリング再起動を実行することにより、すべての所有者に新しいバインドされたトークンを要求するように強制できます。いずれかのアクションを実行する前に、Kubernetes API サーバー Pod の新しいリビジョンがサービスアカウント発行者の変更とともにロールアウトされるのを待ちます。

   a. **cluster Authentication** オブジェクトを編集します。

   ```
   $ oc edit authentications cluster
   ```

   b. **spec.serviceAccountIssuer** フィールドを、必要なサービスアカウント発行者の値に設定します。

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

1. この値は URL である必要があり、バインドされたトークンの受信側はトークンの署名の検証に必要なパブリックキーを取得できます。デフォルトは **https://kubernetes.default.svc** です。

c. 変更を適用するためにファイルを保存します。

d. Kubernetes API サーバー Pod の新しいリビジョンがロールアウトされるまで待ちます。すべてのノードが新規リビジョンに更新されるまで数分かかる場合があります。以下のコマンドを実行します。

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

Kubernetes API サーバーの **NodeInstallerProgressing** 状況条件を確認し、すべてのノードが最新のリビジョンであることを確認します。更新が正常に実行されると、この出力には **AllNodesAtLatestRevision** が表示されます。

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

1. この例では、最新のリビジョン番号は **12** です。

出力に以下のようなメッセージが表示される場合は、更新が進行中です。数分待機した後に再試行します。

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

e. オプション: ノードのローリング再起動を実行するか、クラスター内のすべての Pod を手動で再起動することにより、所有者に新しいバインドされたトークンを要求するように強制します。

- ローリングノード再起動を実行します。

> **警告**
>
> クラスターでカスタムワークロードを実行している場合は、ノードのローリング再起動を実行することは推奨しません。これは、サービスの中断を引き起こす可能性があるためです。代わりに、クラスター内のすべての Pod を手動で再起動します。

ノードを順番に再起動します。次のノードを再起動する前に、ノードが完全に使用可能になるまで待ちます。ノードを再びスケジュール可能としてドレイン、再起動、およびマークする方法については、**ノードの正常な再起動** を参照してください。

- クラスター内のすべての Pod を手動で再起動します。

警告

このコマンドは、すべての namespace で実行されているすべての Pod を削除するため、このコマンドを実行するとサービスが中断します。これらの Pod は削除後に自動的に再起動します。

以下のコマンドを実行します。

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

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

(1) 既存のサービスアカウントへの参照。

(2) 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。

(3) 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。デフォルトは 3600 秒 (1 時間) で、600 秒 (10 分) 以上に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 はトークンのローテーションの試行を開始します。

(4) 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、OpenShift Container Platform 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。

3. 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。

14.3. 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

- **cluster-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
CI6MTY3OTU0MzgzMCwiaWF0IjoxNjc5NTQwMjMwLCJpc3MiOiJodHRwczovL2lzc3VlcjludGV
zdC5jb20iLCJrdWJlcm5ldGVzLmlvIjp7Im5hbWVzcGFjZSI6ImRlZmF1bHQiLCJzZXJ2aWNlYW
Njb3VudCI6eyJuYW1lIjoidGVzdC1zYSIsInVpZCI6ImM3ZjA4MjkwLWIzOTUtNGM4NC04NjI4L
TMzMTM1NTVhNWY1OSJ9fSwibmJmIjoxNjc5NTQwMjMwLCJzdWIiOiJzeXN0ZW06c2Vydmlj
ZWFjY291bnQ6ZGVmYXVsdDp0ZXN0LXNhIn0.WyAOPvh1BFMUl3LNhBCrQeaB5wSynbnCf
ojWuNNPSilT4YvFnKibxwREwmzHpV4LO1xOFZHSi6bXBOmG_o-
m0XNDYL3FrGHd65mymiFyluztxa2lgHVxjw5reIV5ZLgNSol3Y8bJqQqmNg3rtQQWRML2kpJB
XdDHNww0E5XOypmffYkfkadli8lN5QQD-
MhsCbiAF8waCYs8bj6V6Y7uUKTcxee8sCjiRMVtXKjQtooERKm-
CH_p57wxCljIBeM89VdaR51NJGued4hVV5lxvVrYZFu89lBEAq4oyQN_d6N1vBWGXQMyoihn
```

> t_fQjn-NfnlJWk-3NSZDIluDJAv7e-MTEk3geDrHVQKNEzDei2-Un64hSzb-n1g1M0Vn0885wQBQAePC9UlZm8YZlMNk1tq6wIUKQTMv3HPfi5HtBRqVc2eVs0EfMX4-x-PHhPCasJ6qLJWyj6DvyQ08dP4DW_TWZVGvKlmId0hzwpg59TTcLR0iCklSEJgAVEEd13Aa_M0-faD11L3MhUGxw0qxgOsPczdXUsolSISbefs7OKymzFSIkTAn9sDQ8PHMOsuyxsK8vzfrR-E0z7MAeguZ2kaIY7cZqbN6WFy0caWgx46hrKem9vCKALefElRYbCg3hcBmowBcRTOqaFHLNnHghhU1LaRpoFzH7OUarqX9SGQ

関連情報

- ノードを正常に再起動する
- サービスアカウントの作成

第15章 SECURITY CONTEXT CONSTRAINTS の管理

OpenShift Container Platform では、Security Context Constraint (SCC) 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、プラットフォームの Pod をデプロイ時または OpenShift Container Platform のアップグレード時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。

15.1. 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

OpenShift Container Platform のネームスペースに**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 サーバーなどの主要な API グループの起動を管理するために内部 OpenShift Container Platform 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。

15.1.1. 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。オペレーターまたはその他のコンポーネントを OpenShift Container Platform にインストールすると、追加の SCC がインストールされる場合があります。

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、プラットフォームの Pod をデプロイ時または OpenShift Container Platform のアップグレード時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15.1 デフォルトの Security Context Constraints

Security context constraint (SCC)	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。

Security context constraint (SCC)	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。

Security context constraint (SCC)	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。

Security context constraint (SCC)	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。

Security context constraint (SCC)	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。 4.10 以前の OpenShift Container Platform からアップグレードされたクラスターでは、認証済みのユーザーであれば、この SCC を利用できます。アクセスが明示的に付与されない限り、新しい OpenShift Container Platform 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。

15.1.2. 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。

15.1.3. 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。

15.1.4. 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

許可されるボリュームタイプのリストは、新規タイプが OpenShift Container Platform の各リリースと共に追加されるため、網羅的なリストである必要はありません。

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。

15.1.5. 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。

15.1.6. 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。

15.2. 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。

15.3. 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 OpenShift Container Platform によって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 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。

15.4. 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]
```

15.5. 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。

注記

SCC をデフォルト namespace (**default**、**kube-system**、**kube-public**、**openshift-node**、**openshift-infra**、および **openshift**) のいずれかに作成します。これらの namespace は Pod またはサービスの実行に使用しないでください。

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。

15.6. 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 SCC (Security Context Constraints) を通常の API オブジェクトとして管理できます。

注記

SCC を管理できるように **cluster-admin** 権限がなければなりません。

15.6.1. SCC (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"]
```

15.6.2. 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。

注記

アップグレード時にカスタマイズされた SCC を保持するには、デフォルトの SCC の設定を編集しないでください。

15.6.3. SCC (Security Context Constraints) の削除

SCC を削除するには、以下を実行します。

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

注記

デフォルトの SCC を削除する場合、それはクラスターの再起動時に再生成されます。

15.6.4. SCC (Security Context Constraints) の更新

既存 SCC を更新するには、以下を実行します。

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

注記

アップグレード時にカスタマイズされた SCC を保持するには、デフォルトの SCC の設定を編集しないでください。

15.7. 関連情報

- サポート

第16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。

16.1. 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。

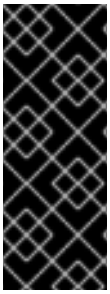
2. セキュリティーコンテキスト制約のコントローラーは、一致する SCC に対して Pod のセキュリティーコンテキストを検証します。

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。

16.2. POD セキュリティー標準とのセキュリティーコンテキスト制約の同期

OpenShift Container Platform には、Kubernetes Pod のセキュリティーアドミッション が含まれます。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。

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の設定に加えて、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** ラベルと **alert** ラベルを namespace に適用するコントローラーが存在します。



重要

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。必要に応じて、他の namespace で Pod セキュリティーアドミッション同期を有効にできます。Operator がユーザー作成の **openshift-*** namespace にインストールされている場合、namespace でクラスターサービスバージョン (CSV) が作成された後、デフォルトで同期がオンになります。

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 のセキュリティーアドミッション **warn** と **aleart** ラベルは、namespace 内で最も特権が高い Pod セキュリティープロファイルに設定され、Pod の作成時に警告と監査ログが発生しないようにします。

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。

16.3. 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

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。これらの namespace には以下が含まれます。

- **default**
- **kube-node-lease**
- **kube-system**
- **kube-public**
- **openshift**
- **openshift-operators** を除く、**openshift-** で始まるすべてのシステム作成 namespace

デフォルトでは、**openshift-** 接頭辞を持つすべての namespace は同期されません。ユーザーが作成した任意の **openshift-*** namespace の同期を有効にすることができます。**openshift-operators** を除き、システムで作成された **openshift-*** namespace の同期を有効にすることはできません。

Operator がユーザー作成の **openshift-*** namespace にインストールされている場合、namespace でクラスターサービスバージョン (CSV) が作成された後、デフォルトで同期がオンになります。同期されたラベルは、namespace のサービスアカウントの権限を継承します。

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
    ```

16.4. 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。

16.4.1. Pod のセキュリティー違反の特定

PodSecurityViolation アラートは、どのワークロードが Pod のセキュリティー違反を引き起こしているかについての詳細を提供しません。影響を受けるワークロードは、Kubernetes API サーバーの監査ログを確認することで特定できます。この手順では、**must-gather** ツールを使用して監査ログを収集し、**pod-security.kubernetes.io/audit-violations** アノテーションを検索します。

前提条件

- **jq** 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アクセスできる。

手順

1. 監査ログを収集するには、次のコマンドを入力します。

```
$ oc adm must-gather -- /usr/bin/gather_audit_logs
```

2. 影響を受けるワークロードの詳細を出力するには、次のコマンドを入力します。

```
$ zgrep -h pod-security.kubernetes.io/audit-violations must-gather.local.
<archive_id>/<image_digest_id>/audit_logs/kube-apiserver/*log.gz \
  | jq -r 'select((.annotations["pod-security.kubernetes.io/audit-violations"] != null) and
(.objectRef.resource=="pods")) | .objectRef.namespace + " " + .objectRef.name' \
  | sort | uniq -c
```

<archive_id> と **<image_digest_id>** を、実際のパス名に置き換えます。

出力例

```
1 test-namespace my-pod
```

16.5. 関連情報

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

第17章 SYSTEM:ADMIN ユーザーの権限の借用

17.1. API の権限借用

OpenShift Container Platform API への要求を、別のユーザーから発信されているかのように設定できます。詳細は、Kubernetes ドキュメントの User impersonation を参照してください。

17.2. SYSTEM:ADMIN ユーザーの権限の借用

クラスター管理者のパーミッションを付与する **system:admin** の権限を借用するユーザーパーミッションを付与することができます。

手順

- **system:admin** の権限を借用するためにユーザーパーミッションを付与するには、以下のコマンドを実行します。

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

ヒント

または、以下の YAML を適用して、**system:admin** の偽装権限を割り当てることができます。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: <any_valid_name>
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: sudoer
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: <username>
```

17.3. SYSTEM:ADMIN グループの権限の借用

system:admin ユーザーにグループ経由でクラスター管理者のパーミッションが付与されている場合、コマンドに **--as=<user> --as-group=<group1> --as-group=<group2>** パラメーターを追加して、関連するグループの権限を借用する必要があります。

手順

- 関連するクラスター管理グループの権限を借用して **system:admin** の権限を借用するためにユーザーパーミッションを付与するには、以下のコマンドを実行します。

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

第18章 LDAP グループの同期

管理者は、グループを使用してユーザーを管理し、権限を変更し、連携を強化できます。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。OpenShift Container Platform はそれらの LDAP レコードを内部 OpenShift Container Platform レコードと同期できるので、グループを 1 つの場所で管理できます。現時点で OpenShift Container Platform は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) を使用してグループと 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 **cluster-admin** 権限を持っている必要があります。

18.1. 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。
- OpenShift Container Platform Group 名を LDAP サーバーのグループにマップする管理者が定義した名前マッピングのリストです。

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。同期操作のエントリーを取得するために昇格した権限が必要となる場合、OpenShift Container Platform はこれを使用します。
3. バインドに使用する任意のパスワードです。同期操作のエントリーを取得するために昇格した権限が必要となる場合、OpenShift Container Platform はこれを使用します。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、OpenShift Container Platform はシステムで信頼される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18.1 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。
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。

LDAP 検索範囲	説明

表18.2 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、OpenShift Container Platform Groups の名前を LDAP サーバーでグループを検出する固有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動作では OpenShift Container Platform Group の名前として指定される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
```

18.1.1. RFC 2307 設定ファイルについて

RFC 2307 スキーマでは、ユーザーとグループエントリー両方の LDAP クエリー定義と内部 OpenShift Container Platform レコードでそれら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、OpenShift Container Platform で作成するグループは (可能な場合) ユーザーまたは管理者に表示されるフィールドに識別名以外の属性を使用する必要があります。たとえば、メールによって OpenShift Container Platform Group のユーザーを識別し、一般名としてグループの名前を使用します。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. OpenShift Container Platform Group レコードでユーザー名として使用される属性です。

18.1.2. 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 OpenShift Container Platform Group レコードでそれら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、OpenShift Container Platform で作成するグループは (可能な場合) ユーザーまたは管理者に表示されるフィールドに識別名以外の属性を使用する必要があります。たとえば、メールによって OpenShift Container Platform Group のユーザーを識別しますが、LDAP サーバーのグループ名でグループの名前を定義します。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. OpenShift Container Platform Group レコードでユーザー名として使用される属性です。
2. メンバーシップ情報を保存するユーザーの属性です。

18.1.3. 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 OpenShift Container Platform Group レコードでそれら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、OpenShift Container Platform で作成するグループは (可能な場合) ユーザーまたは管理者に表示されるフィールドに識別名以外の属性を使用する必要があります。たとえば、メールによって OpenShift Container Platform Group のユーザーを識別し、一般名としてグループの名前を使用します。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、ホワイトリスト/ブラックリストの方法を使用します。

2. グループの名前として使用する属性。
3. OpenShift Container Platform Group レコードでユーザー名として使用される属性です。
4. メンバーシップ情報を保存するユーザーの属性です。

18.2. 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。OpenShift Container Platform 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。

18.2.1. LDAP サーバーの OpenShift Container Platform との同期

LDAP サーバーのすべてのグループを OpenShift Container Platform に同期できます。

前提条件

- 同期設定ファイルを作成します。

手順

- LDAP サーバーからのすべてのグループを OpenShift Container Platform 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ので、OpenShift Container Platform Group レコードを変更するために **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。

18.2.2. OpenShift Container Platform Group の LDAP サーバーとの同期

設定ファイルで指定された LDAP サーバーのグループに対応する OpenShift Container Platform のグループすべてを同期できます。

前提条件

- 同期設定ファイルを作成します。

手順

- OpenShift Container Platform Group 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ので、OpenShift Container Platform Group レコードを変更するために **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。

18.2.3. LDAP サーバーのサブグループの OpenShift Container Platform との同期

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 OpenShift Container Platform 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 LDAP サーバーにあるグループと OpenShift Container Platform にすでにあるグループに適用されます。

前提条件

- 同期設定ファイルを作成します。

手順

- LDAP グループのサブセットを OpenShift Container Platform と同期するには、以下のコマンドを使用します。

```
$ 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ので、OpenShift Container Platform Group レコードを変更するために **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。

18.3. 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 OpenShift Container Platform レコードからグループを削除することを選択できます。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
```

18.4. LDAP グループを自動的に同期する

cron ジョブを設定することにより、LDAP グループを定期的に自動的に同期できます。

前提条件

- **cluster-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。
- LDAP ID プロバイダー (IDP) を設定しました。
  この手順は、**ldap-secret**という名前の LDAP シークレットと **ca-config-map** という名前の設定マップを作成したことを前提としています。

手順

1. cron ジョブを実行するプロジェクトを作成します。

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

   (1) この手順では、**ldap-sync**というプロジェクトを使用します。

2. LDAP ID プロバイダーの設定時に作成したシークレットマップと設定マップを見つけて、この新しいプロジェクトにコピーします。
   シークレットマップと設定マップは**openshift-config**プロジェクトに存在し、新しい**ldap-sync**プロジェクトにコピーする必要があります。

3. サービスアカウントを定義します。

   例: ldap-sync-service-account.yaml

   ■

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

4. サービスアカウントを作成します。

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

5. クラスターのロールを定義します。

例: ldap-sync-cluster-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ldap-group-syncer
rules:
  - apiGroups:
      - ''
      - user.openshift.io
    resources:
      - groups
    verbs:
      - get
      - list
      - create
      - update
```

6. クラスターロールを作成します。

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

7. クラスターロールバインディングを定義して、クラスターロールをサービスアカウントにバインドします。

例: ldap-sync-cluster-role-binding.yaml

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ldap-group-syncer
subjects:
  - kind: ServiceAccount
    name: ldap-group-syncer          (1)
    namespace: ldap-sync
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ldap-group-syncer            (2)
```

(1) この手順の前半で作成したサービスアカウントへの参照。

2 この手順の前半で作成したクラスターのロールへの参照。

8. クラスターロールバインディングを作成します。

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

9. 同期設定ファイルを指定する設定マップを定義します。

例: ldap-sync-config-map.yaml

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
data:
  sync.yaml: |                          1
    kind: LDAPSyncConfig
    apiVersion: v1
    url: ldaps://10.0.0.0:389           2
    insecure: false
    bindDN: cn=admin,dc=example,dc=com      3
    bindPassword:
      file: "/etc/secrets/bindPassword"
    ca: /etc/ldap-ca/ca.crt
    rfc2307:                            4
      groupsQuery:
        baseDN: "ou=groups,dc=example,dc=com"  5
        scope: sub
        filter: "(objectClass=groupOfMembers)"
        derefAliases: never
        pageSize: 0
      groupUIDAttribute: dn
      groupNameAttributes: [ cn ]
      groupMembershipAttributes: [ member ]
      usersQuery:
        baseDN: "ou=users,dc=example,dc=com"  6
        scope: sub
        derefAliases: never
        pageSize: 0
      userUIDAttribute: dn
      userNameAttributes: [ uid ]
      tolerateMemberNotFoundErrors: false
      tolerateMemberOutOfScopeErrors: false
```

1 同期設定ファイルを定義します。

2 URL を指定します。

3 **bindDN** を指定します。

4 この例では、RFC2307 スキーマを使用しています。必要に応じて値を調整します。別のスキーマを使用することもできます。

5. **groupsQuery** の **baseDN** を指定します。
6. **usersQuery** の **baseDN** を指定します。

10. 設定マップを作成します。

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

11. cron ジョブを定義します。

例: ldap-sync-cron-job.yaml

```
kind: CronJob
apiVersion: batch/v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
spec:                                                                  (1)
  schedule: "*/30 * * * *"                                             (2)
  concurrencyPolicy: Forbid
  jobTemplate:
    spec:
      backoffLimit: 0
      ttlSecondsAfterFinished: 1800                                    (3)
      template:
        spec:
          containers:
            - name: ldap-group-sync
              image: "registry.redhat.io/openshift4/ose-cli:latest"
              command:
                - "/bin/bash"
                - "-c"
                - "oc adm groups sync --sync-config=/etc/config/sync.yaml --confirm"  (4)
              volumeMounts:
                - mountPath: "/etc/config"
                  name: "ldap-sync-volume"
                - mountPath: "/etc/secrets"
                  name: "ldap-bind-password"
                - mountPath: "/etc/ldap-ca"
                  name: "ldap-ca"
          volumes:
            - name: "ldap-sync-volume"
              configMap:
                name: "ldap-group-syncer"
            - name: "ldap-bind-password"
              secret:
                secretName: "ldap-secret"                              (5)
            - name: "ldap-ca"
              configMap:
                name: "ca-config-map"                                  (6)
          restartPolicy: "Never"
          terminationGracePeriodSeconds: 30
```

```
        activeDeadlineSeconds: 500
        dnsPolicy: "ClusterFirst"
        serviceAccountName: "ldap-group-syncer"
```

1. cron ジョブの設定を設定します。cron ジョブ設定の詳細については、cron ジョブの作成を参照してください。
2. cron 形式 で指定されるジョブのスケジュール。この例の cron ジョブは 30 分ごとに実行されます。同期の実行にかかる時間を考慮して、必要に応じて周波数を調整します。
3. 完了したジョブを保持する時間 (秒単位)。これは、失敗した以前のジョブを消去して不要なアラートを発生させないように、ジョブスケジュールの期間と同じにする必要があります。詳細については、Kubernetes ドキュメントの TTL-after-finished Controller を参照してください。
4. cron ジョブを実行するための LDAP 同期コマンド。設定マップで定義された同期設定ファイルを渡します。
5. このシークレットは、LDAP IDP が設定されたときに作成されました。
6. この設定マップは、LDAP IDP が設定されたときに作成されました。

12. cron ジョブを作成します。

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

関連情報

- LDAP アイデンティティープロバイダーの設定
- cron ジョブの作成

18.5. 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。

18.5.1. 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 OpenShift Container Platform の Group 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

- 設定ファイルを作成します。

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
```

OpenShift Container Platform は、上記の同期操作の結果として以下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 OpenShift Container Platform Group

```
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 OpenShift Container Platform Group と LDAP サーバーが最後に同期された時間です。ISO 6801 形式を使用します。
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。

18.5.2. 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. 固有の識別子がユーザー定義の名前マッピングに存在しない場合に OpenShift Container Platform Group に名前を付けるための属性です。
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

- 設定ファイルを作成します。

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
```

OpenShift Container Platform は、上記の同期操作の結果として以下のグループレコードを作成します。

rfc2307_config_user_defined.yaml ファイルを使用して作成される OpenShift Container Platform Group

```
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。

18.5.3. 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 OpenShift Container Platform Group に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

- 設定ファイルを作成します。

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
```

OpenShift Container Platform は、上記の同期操作の結果として以下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 OpenShift Container Platform Group

```
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。

18.5.4. 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

- 設定ファイルを作成します。

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
```

OpenShift Container Platform は、上記の同期操作の結果として以下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 OpenShift Container Platform Group

```
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 OpenShift Container Platform Group と LDAP サーバーが最後に同期された時間です。ISO 6801 形式を使用します。
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。

18.5.5. 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

- 設定ファイルを作成します。

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
```

OpenShift Container Platform は、上記の同期操作の結果として以下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 OpenShift Container Platform Group

```
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 OpenShift Container Platform Group と LDAP サーバーが最後に同期された時間です。ISO 6801 形式を使用します。
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。

18.5.5.1. LDAP のネスト化されたメンバーシップ同期の例

OpenShift Container Platform の Group はネスト化しません。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 OpenShift Container Platform の Group 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。

Active Directory を使用してネスト化されたグループを同期するには、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 OpenShift Container Platform Group レコードでそれら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、OpenShift Container Platform で作成するグループは (可能な場合) ユーザーまたは管理者に表示されるフィールドに識別名以外の属性を使用する必要があります。たとえば、メールによって OpenShift Container Platform Group のユーザーを識別し、一般名としてグループの名前を使用します。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. OpenShift Container Platform Group レコードでユーザー名として使用される属性です。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

- 設定ファイルを作成します。

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。

OpenShift Container Platform は、上記の同期操作の結果として以下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た OpenShift Container Platform 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 OpenShift Container Platform Group と LDAP サーバーが最後に同期された時間です。ISO 6801 形式を使用します。
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。

18.6. 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。

18.6.1. 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
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

名前	説明	スキーマ
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 の OpenShift Container Platform Group 名への任意の直接マッピングです。	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
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

18.6.2. 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

18.6.3. 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
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

名前	説明	スキーマ
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

18.6.4. 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
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が OpenShift Container Platform Group に使用する名前として解釈されるかを定義します。	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

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
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が順番に OpenShift Container Platform ユーザー名として使われるかを定義します。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。OpenShift Container Platform Group レコードでユーザー名として使用される属性です。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
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

名前	説明	スキーマ
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

18.6.5. 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
userNameAttributes	LDAP ユーザーエントリーのどの属性が OpenShift Container Platform ユーザー名として解釈されるかを定義します。OpenShift Container Platform Group レコードでユーザー名として使用される属性です。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

18.6.6. 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が OpenShift Container Platform ユーザー名として解釈されるかを定義します。OpenShift Container Platform Group レコードでユーザー名として使用される属性です。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
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が OpenShift Container Platform Group に使用する名前として解釈されるかを定義します。	文字列配列

第19章 クラウドプロバイダーの認証情報の管理

19.1. CLOUD CREDENTIAL OPERATOR について

Cloud Credential Operator (CCO) は、クラウドプロバイダーの認証情報をカスタムリソース定義 (CRD) として管理します。CCO は **CredentialsRequest** カスタムリソース (CR) で同期し、OpenShift Container Platform コンポーネントが、クラスターの実行に必要な特定のパーミッションと共にクラウドプロバイダーの認証情報を要求できるようにします。

install-config.yaml ファイルで **credentialsMode** パラメーターに異なる値を設定すると、CCO は複数の異なるモードで動作するように設定できます。モードが指定されていない場合や、**credentialsMode** パラメーターが空の文字列 (**""**) に設定されている場合、CCO はデフォルトモードで動作します。

19.1.1. モード

install-config.yaml ファイルに **credentialsMode** パラメーターの異なる値を設定することで、CCO を **mint**、**passthrough**、または **manual** モードで動作するように設定できます。これらのオプションにより、CCO がクラウド認証情報を使用してクラスターで **CredentialsRequest** CR を処理し、CCO を組織のセキュリティー要件に対応するように設定する方法において透明性と柔軟性が提供されます。すべてのクラウドプロバイダーですべての CCO モードがサポートされている訳ではありません。

- **Mint**: mint モードでは、CCO は提供される管理レベルのクラウド認証情報を使用して、必要となる特定のパーミッションのみでクラスター内のコンポーネントの新規の認証情報を作成します。
- **Passthrough**: passthrough モードでは、CCO はクラウド認証情報を要求するコンポーネントに、指定されたクラウド認証情報を渡します。
- **Manual**: manual モードでは、ユーザーは CCO の代わりにクラウド認証情報を管理します。
  - **AWS Security Token Service による手動** 手動モードでは、AWS クラスターを設定して、Amazon Web Services Security Token Service (AWS STS) を使用できます。この設定では、CCO は異なるコンポーネントに一時的な認証情報を使用します。
  - **GCP ワークロード ID を使用した手動** 手動モードでは、GCP ワークロード ID を使用するように GCP クラスターを設定できます。この設定では、CCO は異なるコンポーネントに一時的な認証情報を使用します。

表19.1 CCO モードのサポートマトリックス

クラウドプロバイダー	Mint	passthrough	手動
Alibaba Cloud			X
Amazon Web Services (AWS)	X	X	X
Microsoft Azure		X [1]	X
Google Cloud Platform (GCP)	X	X	X
IBM Cloud			X

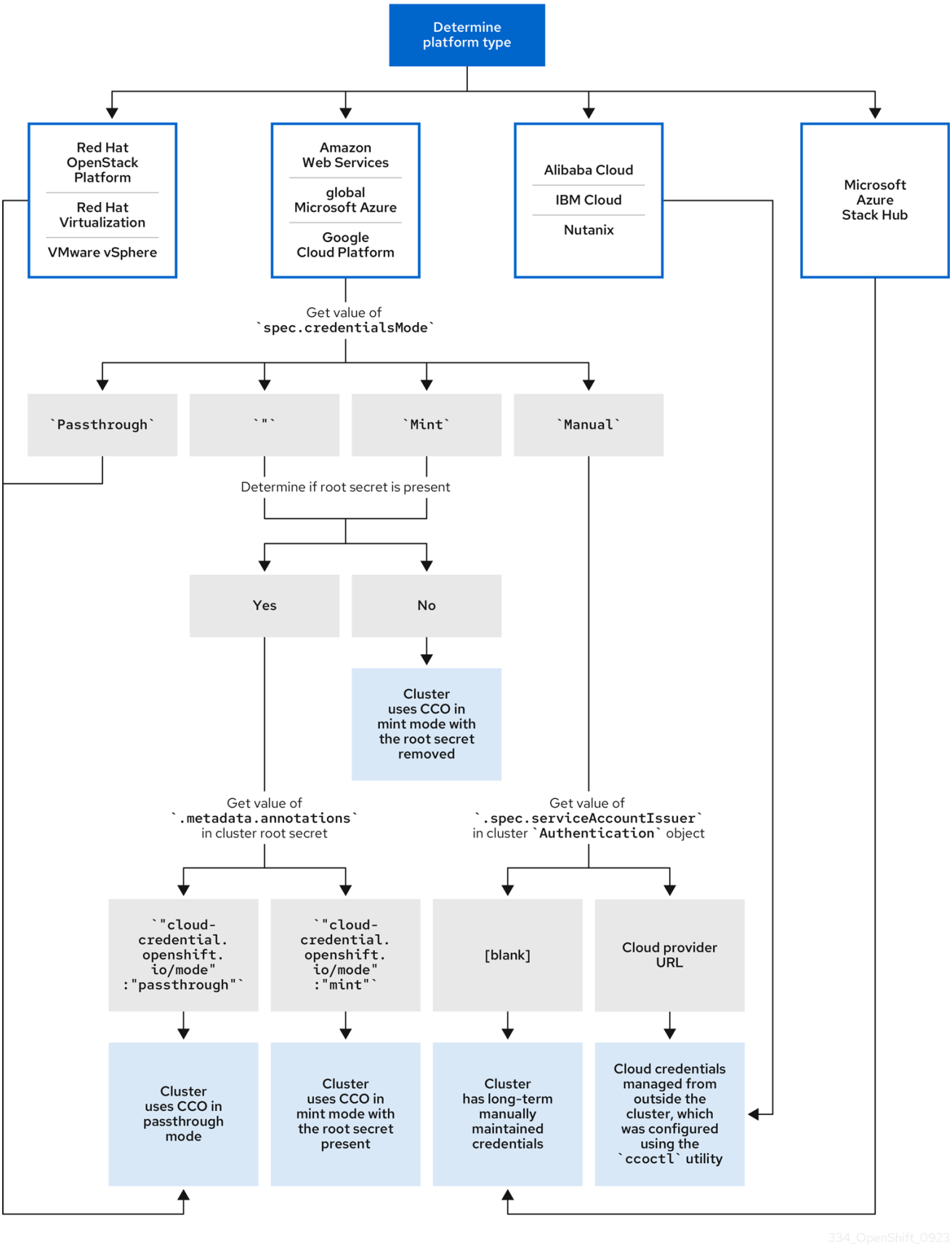
クラウドプロバイダー	Mint	passthrough	手動
Nutanix			X
Red Hat OpenStack Platform (RHOSP)		X	
Red Hat Virtualization (RHV)		X	
VMware vSphere		X	

1. 手動モードは、Microsoft Azure Stack Hub でサポートされている唯一の CCO 設定です。

19.1.2. Cloud Credential Operator モードの決定

複数のモードでの CCO の使用をサポートするプラットフォームの場合、Web コンソールまたは CLI を使用して、CCO がどのモードを使用するように設定されているかを判断できます。

図19.1 CCO 設定の決定



19.1.2.1. Web コンソールを使用した Cloud Credential Operator モードの判別

Cloud Credential Operator (CCO) がどのモードを使用するように設定されているかは、Web コンソールを使用して判別できます。

注記

複数の CCO モードをサポートするのは、Amazon Web Services (AWS)、グローバル Microsoft Azure、および Google Cloud Platform (GCP) クラスターのみです。

前提条件

- クラスター管理者パーミッションを持つ OpenShift Container Platform アカウントにアクセスできる。

手順

1. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform Web コンソールにログインします。
2. **Administration → Cluster Settings** に移動します。
3. **Cluster Settings** ページで、**Configuration** タブを選択します。
4. **Configuration resource** で **CloudCredential** を選択します。
5. **CloudCredential details** ページで、**YAML** タブを選択します。
6. YAML ブロックで、**spec.credentialsMode** の値を確認します。次の値が可能ですが、すべてのプラットフォームですべてがサポートされているわけではありません。
   - **''**: CCO はデフォルトモードで動作しています。この設定では、CCO は、インストール中に提供されたクレデンシャルに応じて、ミントモードまたはパススルーモードで動作します。
   - **Mint**: CCO はミントモードで動作しています。
   - **Passthrough**: CCO はパススルーモードで動作しています。
   - **Manual**: CCO は手動モードで動作します。

   重要

   spec.credentialsMode が **''**、**Mint**、または **Manual** である AWS または GCP クラスターの特定の設定を特定するには、さらに調査する必要があります。

   AWS および GCP クラスターは、ルートシークレットが削除されたミントモードの使用をサポートします。

   手動モードを使用する AWS または GCP クラスターは、AWS Security Token Service (STS) または GCP Workload Identity を使用して、クラスターの外部からクラウド認証情報を作成および管理するように設定されている場合があります。クラスター **Authentication** オブジェクトを調べることで、クラスターがこの戦略を使用しているかどうかを判断できます。

7. デフォルト (**''**) のみを使用する AWS または GCP クラスター: クラスターがミントモードまたはパススルーモードで動作しているかどうかを判断するには、クラスタールートシークレットのアノテーションを調べます。
   a. **Workloads → Secrets** に移動し、クラウドプロバイダーのルートシークレットを探します。

注記

Project ドロップダウンが **All Projects** に設定されていることを確認します。

プラットフォーム	シークレット名
AWS	**aws-creds**
GCP	**gcp-credentials**

b. クラスターが使用している CCO モードを表示するには、**Annotations** で **1 annotation** をクリックし、value フィールドを確認します。以下の値が可能です。

- **Mint**: CCO はミントモードで動作しています。
- **Passthrough**: CCO はパススルーモードで動作しています。

クラスターが mint モードを使用している場合、クラスターがルートシークレットなしで動作しているかどうかを判断することもできます。

8. mint モードのみを使用する AWS または GCP クラスター: クラスターがルートシークレットなしで動作しているかどうかを判断するには、**Workloads → Secrets** に移動し、クラウドプロバイダーのルートシークレットを探します。

注記

Project ドロップダウンが **All Projects** に設定されていることを確認します。

プラットフォーム	シークレット名
AWS	**aws-creds**
GCP	**gcp-credentials**

- これらの値のいずれかが表示される場合、クラスターはルートシークレットが存在するミントモードまたはパススルーモードを使用しています。
- これらの値が表示されない場合、クラスターはルートシークレットが削除されたミントモードで CCO を使用しています。

9. 手動モードのみを使用する AWS または GCP クラスター: クラスターがクラスターの外部からクラウド認証情報を作成および管理するように設定されているかどうかを判断するには、クラスター **Authentication** オブジェクトの YAML 値を確認する必要があります。

a. **Administration → Cluster Settings** に移動します。

b. **Cluster Settings** ページで、**Configuration** タブを選択します。

c. **Configuration resource** で **Authentication** を選択します。

d. **Authentication details** ページで、**YAML** タブを選択します。

e. YAML ブロックで、**.spec.serviceAccountIssuer** パラメーターの値を確認します。

- クラウドプロバイダーに関連付けられている URL を含む値は、CCO が AWS STS または GCP Workload Identity で手動モードを使用して、クラスターの外部からクラウド認証情報を作成および管理していることを示します。これらのクラスターは、**ccoctl** ユーティリティーを使用して設定されます。

- 空の値 (**''**) は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティーを使用して設定されていないことを示します。

19.1.2.2. CLI を使用した Cloud Credential Operator モードの判別

CLI を使用して、Cloud Credential Operator (CCO) が使用するように設定されているモードを判別できます。

注記

複数の CCO モードをサポートするのは、Amazon Web Services (AWS)、グローバル Microsoft Azure、および Google Cloud Platform (GCP) クラスターのみです。

前提条件

- クラスター管理者パーミッションを持つ OpenShift Container Platform アカウントにアクセスできる。

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

手順

1. **cluster-admin** ロールを持つユーザーとしてクラスターの **oc** にログインします。

2. CCO が使用するように設定されているモードを確認するには、次のコマンドを入力します。

```
$ oc get cloudcredentials cluster \
  -o=jsonpath={.spec.credentialsMode}
```

すべてのプラットフォームですべてがサポートされているわけではありませんが、次の出力値が可能です。

- **''**: CCO はデフォルトモードで動作しています。この設定では、CCO は、インストール中に提供されたクレデンシャルに応じて、ミントモードまたはパススルーモードで動作します。

- **Mint**: CCO はミントモードで動作しています。

- **Passthrough**: CCO はパススルーモードで動作しています。

- **Manual**: CCO は手動モードで動作します。

重要

spec.credentialsMode が **''**、**Mint**、または **Manual** である AWS または GCP クラスターの特定の設定を特定するには、さらに調査する必要があります。

AWS および GCP クラスターは、ルートシークレットが削除されたミントモードの使用をサポートします。

手動モードを使用する AWS または GCP クラスターは、AWS Security Token Service (STS) または GCP Workload Identity を使用して、クラスターの外部からクラウド認証情報を作成および管理するように設定されている場合があります。クラスター **Authentication** オブジェクトを調べることで、クラスターがこの戦略を使用しているかどうかを判断できます。

3. デフォルト (**''**) のみを使用する AWS または GCP クラスター: クラスターがミントモードまたはパススルーモードで動作しているかどうかを判断するには、次のコマンドを実行します。

```
$ oc get secret <secret_name> \
  -n kube-system \
  -o jsonpath \
  --template '{ .metadata.annotations }'
```

<secret_name> は、AWS の場合は **aws-creds**、GCP の場合は **gcp-credentials** です。

このコマンドは、クラスタールートシークレットオブジェクトの **.metadata.annotations** パラメーターの値を表示します。以下の出力値を使用できます。

- **Mint**: CCO はミントモードで動作しています。
- **Passthrough**: CCO はパススルーモードで動作しています。

クラスターが mint モードを使用している場合、クラスターがルートシークレットなしで動作しているかどうかを判断することもできます。

4. mint モードのみを使用する AWS または GCP クラスター: クラスターがルートシークレットなしで動作しているかどうかを判断するには、次のコマンドを実行します。

```
$ oc get secret <secret_name> \
  -n=kube-system
```

<secret_name> は、AWS の場合は **aws-creds**、GCP の場合は **gcp-credentials** です。

ルートシークレットが存在する場合、このコマンドの出力はシークレットに関する情報を返します。エラーは、ルートシークレットがクラスターに存在しないことを示します。

5. 手動モードのみを使用する AWS または GCP クラスター: クラスターがクラスターの外部からクラウド認証情報を作成および管理するように設定されているかどうかを確認するには、次のコマンドを実行します。

```
$ oc get authentication cluster \
  -o jsonpath \
  --template='{ .spec.serviceAccountIssuer }'
```

このコマンドは、クラスター **Authentication** オブジェクトの **.spec.serviceAccountIssuer** パラメーターの値を表示します。

- クラウドプロバイダーに関連付けられている URL の出力は、CCO が AWS STS または GCP Workload Identity で手動モードを使用して、クラスターの外部からクラウド認証情報を作成および管理していることを示しています。これらのクラスターは、**ccoctl** ユーティリティーを使用して設定されます。

- 空の出力は、クラスターが手動モードで CCO を使用しているが、**ccoctl** ユーティリティーを使用して設定されていないことを示します。

19.1.3. デフォルト動作

複数のモードがサポートされるプラットフォーム (AWS、Azure、および GCP) の場合、CCO がデフォルトモードで動作する際に、これは指定される認証情報を動的にチェックし、**CredentialsRequest** CR を処理するのに十分なモードを判別します。

デフォルトで、CCO は推奨される操作モードの mint モードに十分な認証情報があるかどうかを判別し、これらの認証情報を使用してクラスター内のコンポーネントの適切な認証情報を作成します。mint モードに十分な認証情報がない場合は、passthrough モードに十分な認証情報があるかどうかを判別します。passthrough モードに十分な認証情報がない場合、CCO は **CredentialsRequest** CR を適切に処理できません。

インストール時に提供された認証情報が不十分であると判別される場合、インストールは失敗します。AWS では、インストーラーはプロセスの早期の段階で失敗し、どの必須パーミッションが欠落しているかを示します。他のプロバイダーは、エラーが発生するまでエラーの原因についての具体的な情報を提供しない場合があります。

認証情報が正常なインストールの後に変更され、CCO が新規の認証情報が不十分であると判別する場合に、CCO は新規の **CredentialsRequest** CR に条件を追加し、認証情報が不十分であるためにそれらを処理できないことを示唆します。

不十分な認証情報についての問題を解決するために、適切なパーミッションで認証情報を指定します。エラーがインストール時に発生した場合は、再度インストールを試行します。新規の **CredentialsRequest** CR 関連の問題については、CCO が再び CR の処理を試行するのを待機します。別の方法として、AWS、Azure、および GCP の IAM を手動で作成できます。

19.1.4. 関連情報

- Cloud Credential Operator の Cloud Credential Operator リファレンスページ

19.2. MINT モードの使用

Mint モードは、Amazon Web Services (AWS) と Google Cloud Platform (GCP) でサポートされています。

Mint モードは、サポートされているプラットフォームのデフォルトモードです。このモードでは、Cloud Credential Operator (CCO) は提供される管理者レベルのクラウド認証情報を使用して、必要となる特定のパーミッションのみでクラスター内のコンポーネントの新規の認証情報を作成します。

インストール後に認証情報が削除されない場合、これは CCO によって保存され、クラスター内のコンポーネントの **CredentialsRequest** CR を処理し、必要な特定のパーミッションのみでそれぞれの新規の認証情報を作成するために使用されます。mint モードでクラウド認証情報を継続的に調整することで、アップグレードなどの追加の認証情報またはパーミッションを必要とするアクションを続行できます。

Mint モードでは、管理者レベルのクレデンシャルがクラスター **kube-system** namespace に格納されます。このアプローチが組織のセキュリティー要件を満たしていない場合は、AWS または GCP の **kube-system プロジェクトに管理者レベルのシークレットを保存する代わりの方法** をご覧ください。

19.2.1. mint モードのパーミッション要件

mint モードで CCO を使用する場合、指定する認証情報が OpenShift Container Platform を実行し、インストールしているクラウドの各種要件を満たしていることを確認してください。指定される認証情報が mint モードで不十分な場合、CCO は IAM ユーザーを作成できません。

19.2.1.1. Amazon Web Services (AWS) パーミッション

AWS で mint モードに指定する認証情報には以下のパーミッションが必要です。

- **iam:CreateAccessKey**
- **iam:CreateUser**
- **iam:DeleteAccessKey**
- **iam:DeleteUser**
- **iam:DeleteUserPolicy**
- **iam:GetUser**
- **iam:GetUserPolicy**
- **iam:ListAccessKeys**
- **iam:PutUserPolicy**
- **iam:TagUser**
- **iam:SimulatePrincipalPolicy**

19.2.1.2. Google Cloud Platform (GCP) パーミッション

GCP の mint モードに指定する認証情報には以下のパーミッションが必要です。

- **resourcemanager.projects.get**
- **serviceusage.services.list**
- **iam.serviceAccountKeys.create**
- **iam.serviceAccountKeys.delete**
- **iam.serviceAccounts.create**
- **iam.serviceAccounts.delete**
- **iam.serviceAccounts.get**
- **iam.roles.get**
- **resourcemanager.projects.getIamPolicy**

- **resourcemanager.projects.setIamPolicy**

19.2.2. 管理者の認証情報のルートシークレット形式

各クラウドプロバイダーは、**kube-system** namespace の認証情報ルートシークレットを使用します。これは、すべての認証情報要求を満たし、それぞれのシークレットを作成するために使用されます。これは、**mint モード** で新規の認証情報を作成するか、**passthrough モード** で認証情報 root シークレットをコピーして実行します。

シークレットの形式はクラウドごとに異なり、それぞれの **CredentialsRequest** シークレットにも使用されます。

Amazon Web Services (AWS) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <base64-encoded_access_key_id>
  aws_secret_access_key: <base64-encoded_secret_access_key>
```

Google Cloud Platform (GCP) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

19.2.3. 管理者レベルの認証情報の削除またはローテーション機能を持つ mint モード

現時点で、このモードは AWS および GCP でのみサポートされます。

このモードでは、ユーザーは通常の mint モードと同様に管理者レベルの認証情報を使用して OpenShift Container Platform をインストールします。ただし、このプロセスはクラスターのインストール後の管理者レベルの認証情報シークレットを削除します。

管理者は、Cloud Credential Operator に読み取り専用の認証情報について独自の要求を行わせることができます。これにより、すべての **CredentialsRequest** オブジェクトに必要なパーミッションがあることの確認が可能になります。そのため、いずれかの変更が必要にならない限り、管理者レベルの認証情報は必要になりません。関連付けられた認証情報が削除された後に、必要な場合は、これは基礎となるクラウドで破棄するか、非アクティブにできます。

注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

管理者レベルの認証情報はクラスターに永続的に保存されません。

これらの手順を実行するには、短い期間にクラスターでの管理者レベルの認証情報が必要になります。また、アップグレードごとに管理者レベルの認証情報を使用してシークレットを手動で再インストールする必要があります。

19.2.3.1. クラウドプロバイダーの認証情報の手動によるローテーション

クラウドプロバイダーの認証情報が何らかの理由で変更される場合、クラウドプロバイダーの認証情報の管理に Cloud Credential Operator (CCO) が使用するシークレットを手動で更新する必要があります。

クラウド認証情報をローテーションするプロセスは、CCO を使用するように設定されているモードによって変わります。mint モードを使用しているクラスターの認証情報をローテーションした後に、削除された認証情報で作成されたコンポーネントの認証情報は手動で削除する必要があります。

前提条件

- クラスターは、使用している CCO モードでのクラウド認証情報の手動ローテーションをサポートするプラットフォームにインストールされている。
  - mint モードについては、Amazon Web Services (AWS) および Google Cloud Platform (GCP) がサポートされます。
- クラウドプロバイダーとのインターフェイスに使用される認証情報を変更している。
- 新規認証情報には、モードの CCO がクラスターで使用されるように設定するのに十分なパーミッションがある。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Workloads → Secrets** に移動します。
2. **Secrets** ページの表で、クラウドプロバイダーのルートシークレットを見つけます。

プラットフォーム	シークレット名
AWS	**aws-creds**
GCP	**gcp-credentials**

3. シークレットと同じ行にある **Options** メニュー ⋮ をクリックし、**Edit Secret** を選択します。
4. **Value** フィールドの内容を記録します。この情報を使用して、認証情報の更新後に値が異なることを確認できます。
5. **Value** フィールドのテキストをクラウドプロバイダーの新規の認証情報で更新し、**Save** をクリックします。
6. 個々の **CredentialsRequest** オブジェクトによって参照される各コンポーネントシークレットを削除します。
   a. **cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインします。

b. 参照されたすべてのコンポーネントシークレットの名前および namespace を取得します。

```
$ oc -n openshift-cloud-credential-operator get CredentialsRequest \
  -o json | jq -r '.items[] | select (.spec.providerSpec.kind=="<provider_spec>") |
.spec.secretRef'
```

ここで、**<provider_spec>** はクラウドプロバイダーの対応する値になります。

- AWS: **AWSProviderSpec**
- GCP: **GCPProviderSpec**

AWS の部分的なサンプル出力

```
{
  "name": "ebs-cloud-credentials",
  "namespace": "openshift-cluster-csi-drivers"
}
{
  "name": "cloud-credential-operator-iam-ro-creds",
  "namespace": "openshift-cloud-credential-operator"
}
```

c. 参照されるコンポーネントの各シークレットを削除します。

```
$ oc delete secret <secret_name> \ (1)
  -n <secret_namespace> (2)
```

1. シークレットの名前を指定します。
2. シークレットを含む namespace を指定します。

AWS シークレットの削除例

```
$ oc delete secret ebs-cloud-credentials -n openshift-cluster-csi-drivers
```

プロバイダーコンソールから認証情報を手動で削除する必要はありません。参照されるコンポーネントのシークレットを削除すると、CCO はプラットフォームから既存の認証情報を削除し、新規の認証情報を作成します。

検証

認証情報が変更されたことを確認するには、以下を実行します。

1. Web コンソールの **Administrator** パースペクティブで、**Workloads → Secrets** に移動します。
2. **Value** フィールドの内容が変更されていることを確認します。

19.2.3.2. クラウドプロバイダーの認証情報の削除

Cloud Credential Operator (CCO) を mint モードで使用して OpenShift Container Platform クラスターをインストールした後に、クラスターの **kube-system** namespace から管理者レベルの認証情報シークレットを削除できます。管理者レベルの認証情報は、アップグレードなどの昇格されたパーミッションを必要とする変更時にのみ必要です。

注記

z-stream 以外のアップグレードの前に、認証情報のシークレットを管理者レベルの認証情報と共に元に戻す必要があります。認証情報が存在しない場合は、アップグレードがブロックされる可能性があります。

前提条件

- クラスターが、CCO からのクラウド認証情報の削除をサポートするプラットフォームにインストールされている。サポート対象プラットフォームは AWS および GCP。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Workloads → Secrets** に移動します。
2. **Secrets** ページの表で、クラウドプロバイダーのルートシークレットを見つけます。

プラットフォーム	シークレット名
AWS	**aws-creds**
GCP	**gcp-credentials**

3. シークレットと同じ行にある **Options** メニュー ⋮ をクリックし、**Delete Secret** を選択します。

19.2.4. 関連情報

- AWS についての 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法
- GCP についての 管理者レベルのシークレットを kube-system プロジェクトに保存する代替方法

19.3. PASSTHROUGH モードの使用

passthrough モードは、Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Platform (GCP)、Red Hat OpenStack Platform (RHOSP)、Red Hat Virtualization (RHV)、および VMware vSphere でサポートされます。

passthrough モードでは、Cloud Credential Operator (CCO) は提供されるクラウド認証情報を、コンポーネントを要求するコンポーネントに渡します。認証情報には、インストールを実行し、クラスター内のコンポーネントで必要な操作を完了するためのパーミッションが必要ですが、認証情報を新たに作成する必要はありません。CCO は、passthrough モードで、追加の制限されたスコープの認証情報の作成を試行しません。

注記

手動モード は、Microsoft Azure Stack Hub でサポートされている唯一の CCO 設定です。

19.3.1. passthrough モードのパーミッション要件

passthrough モードで CCO を使用する場合、指定する認証情報が OpenShift Container Platform を実行し、インストールしているクラウドの各種要件を満たしていることを確認してください。CCO が **CredentialsRequest** CR を作成するコンポーネントに渡す指定された認証情報が不十分な場合に、そのコンポーネントは、パーミッションがない API の呼び出しを試行する際にエラーを報告します。

19.3.1.1. Amazon Web Services (AWS) パーミッション

AWS の passthrough モードに指定する認証情報には、実行し、インストールしている OpenShift Container Platform のバージョンで必要なすべての **CredentialsRequest** CR に必要なすべての必須パーミッションがなければなりません。

必要な **CredentialsRequest** CR を見つけるには、AWS の IAM の手動作成 について参照してください。

19.3.1.2. Microsoft Azure パーミッション

Azure の passthrough モードに指定する認証情報には、実行し、インストールしている OpenShift Container Platform のバージョンで必要なすべての **CredentialsRequest** CR に必要なすべての必須パーミッションがなければなりません。

必要な **CredentialsRequest** CR を見つけるには、Azure の IAM の手動作成 について参照してください。

19.3.1.3. Google Cloud Platform (GCP) パーミッション

GCP の passthrough モードに指定する認証情報には、実行し、インストールしている OpenShift Container Platform のバージョンで必要なすべての **CredentialsRequest** CR に必要なすべての必須パーミッションがなければなりません。

必要な **CredentialsRequest** CR を見つけるには、GCP の IAM の手動作成 について参照してください。

19.3.1.4. Red Hat OpenStack Platform (RHOSP) パーミッション

OpenShift Container Platform クラスターを RHOSP にインストールするには、CCO では **member** ユーザーロールのパーミッションと共に認証情報が必要になります。

19.3.1.5. Red Hat Virtualization (RHV) パーミッション

OpenShift Container Platform クラスターを RHV にインストールするには、CCO には、以下の権限と共に認証情報が必要になります。

- **DiskOperator**
- **DiskCreator**
- **UserTemplateBasedVm**
- **TemplateOwner**
- **TemplateCreator**

- OpenShift Container Platform デプロイメントにターゲットが設定される特定クラスターの **ClusterAdmin**

19.3.1.6. VMware vSphere パーミッション

OpenShift Container Platform クラスターを VMware vSphere にインストールするには、CCO には以下の vSphere 権限と共に認証情報が必要になります。

表19.2 必要な vSphere 権限

カテゴリー	権限
データストア	**領域の割り当て**
フォルダー	**フォルダーの作成**、**フォルダーの削除**
vSphere タグ	すべての権限
ネットワーク	**ネットワークの割り当て**
リソース	**仮想マシンのリソースプールへの割り当て**
プロファイル駆動型ストレージ	すべての権限
vApp	すべての権限
仮想マシン	すべての権限

19.3.2. 管理者の認証情報のルートシークレット形式

各クラウドプロバイダーは、**kube-system** namespace の認証情報ルートシークレットを使用します。これは、すべての認証情報要求を満たし、それぞれのシークレットを作成するために使用されます。これは、**mint モード** で新規の認証情報を作成するか、**passthrough モード** で認証情報 root シークレットをコピーして実行します。

シークレットの形式はクラウドごとに異なり、それぞれの **CredentialsRequest** シークレットにも使用されます。

Amazon Web Services (AWS) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: aws-creds
stringData:
  aws_access_key_id: <base64-encoded_access_key_id>
  aws_secret_access_key: <base64-encoded_secret_access_key>
```

Microsoft Azure シークレットの形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: azure-credentials
stringData:
  azure_subscription_id: <base64-encoded_subscription_id>
  azure_client_id: <base64-encoded_client_id>
  azure_client_secret: <base64-encoded_client_secret>
  azure_tenant_id: <base64-encoded_tenant_id>
  azure_resource_prefix: <base64-encoded_resource_prefix>
  azure_resourcegroup: <base64-encoded_resource_group>
  azure_region: <base64-encoded_region>
```

Microsoft Azure では、認証情報シークレット形式には、それぞれのクラスターのインストールにランダムに生成されるクラスターのインフラストラクチャー ID が含まれる必要のある 2 つのプロパティーがあります。この値は、マニフェストの作成後に確認できます。

```
$ cat .openshift_install_state.json | jq '."*installconfig.ClusterID".InfraID' -r
```

出力例

```
mycluster-2mpcn
```

この値は、以下のようにシークレットデータで使用されます。

```
azure_resource_prefix: mycluster-2mpcn
azure_resourcegroup: mycluster-2mpcn-rg
```

Google Cloud Platform (GCP) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: gcp-credentials
stringData:
  service_account.json: <base64-encoded_service_account>
```

Red Hat OpenStack Platform (RHOSP) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: openstack-credentials
data:
  clouds.yaml: <base64-encoded_cloud_creds>
  clouds.conf: <base64-encoded_cloud_creds_init>
```

Red Hat Virtualization (RHV) シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: ovirt-credentials
data:
  ovirt_url: <base64-encoded_url>
  ovirt_username: <base64-encoded_username>
  ovirt_password: <base64-encoded_password>
  ovirt_insecure: <base64-encoded_insecure>
  ovirt_ca_bundle: <base64-encoded_ca_bundle>
```

VMware vSphere シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: kube-system
  name: vsphere-creds
data:
 vsphere.openshift.example.com.username: <base64-encoded_username>
 vsphere.openshift.example.com.password: <base64-encoded_password>
```

19.3.3. passthrough モードの認証情報のメンテナンス

CredentialsRequest CR がクラスターのアップグレード時に変更される場合、各種要件を満たすために passthrough モードの認証情報を手動で更新する必要があります。アップグレード時の認証情報の問題を回避するには、アップグレードの前に、新規バージョンの OpenShift Container Platform のリリースイメージで **CredentialsRequest** CR を確認します。クラウドプロバイダーに必要な **CredentialsRequest** CR を見つけるには、AWS、Azure、または GCP の **IAM の手動作成**について参照してください。

19.3.3.1. クラウドプロバイダーの認証情報の手動によるローテーション

クラウドプロバイダーの認証情報が何らかの理由で変更される場合、クラウドプロバイダーの認証情報の管理に Cloud Credential Operator (CCO) が使用するシークレットを手動で更新する必要があります。

クラウド認証情報をローテーションするプロセスは、CCO を使用するように設定されているモードによって変わります。mint モードを使用しているクラスターの認証情報をローテーションした後に、削除された認証情報で作成されたコンポーネントの認証情報は手動で削除する必要があります。

前提条件

- クラスターは、使用している CCO モードでのクラウド認証情報の手動ローテーションをサポートするプラットフォームにインストールされている。
  - passthrough モードは、Amazon Web Services (AWS)、Microsoft Azure、Google Cloud Platform (GCP)、Red Hat OpenStack Platform (RHOSP)、Red Hat Virtualization (RHV)、および VMware vSphere でサポートされます。
- クラウドプロバイダーとのインターフェイスに使用される認証情報を変更している。

- 新規認証情報には、モードの CCO がクラスターで使用されるように設定するのに十分なパーミッションがある。

手順

1. Web コンソールの **Administrator** パースペクティブで、**Workloads → Secrets** に移動します。
2. **Secrets** ページの表で、クラウドプロバイダーのルートシークレットを見つけます。

プラットフォーム	シークレット名
AWS	**aws-creds**
Azure	**azure-credentials**
GCP	**gcp-credentials**
RHOSP	**openstack-credentials**
RHV	**ovirt-credentials**
VMware vSphere	**vsphere-creds**

3. シークレットと同じ行にある **Options** メニュー ⋮ をクリックし、**Edit Secret** を選択します。
4. **Value** フィールドの内容を記録します。この情報を使用して、認証情報の更新後に値が異なることを確認できます。
5. **Value** フィールドのテキストをクラウドプロバイダーの新規の認証情報で更新し、**Save** をクリックします。
6. vSphere CSI Driver Operator が有効になっていない vSphere クラスターの認証情報を更新する場合は、Kubernetes コントローラーマネージャーを強制的にロールアウトして更新された認証情報を適用する必要があります。

注記

vSphere CSI Driver Operator が有効になっている場合、この手順は不要です。

更新された vSphere 認証情報を適用するには、**cluster-admin** ロールを持つユーザーとして OpenShift Container Platform CLI にログインし、以下のコマンドを実行します。

```
$ oc patch kubecontrollermanager cluster \
  -p='{"spec": {"forceRedeploymentReason": "recovery-'"$( date )"'"}}' \
  --type=merge
```

認証情報がロールアウトされている間、Kubernetes Controller Manager Operator のステータスは **Progressing=true** を報告します。ステータスを表示するには、次のコマンドを実行します。

```
$ oc get co kube-controller-manager
```

検証

認証情報が変更されたことを確認するには、以下を実行します。

1. Web コンソールの **Administrator** パースペクティブで、**Workloads → Secrets** に移動します。
2. **Value** フィールドの内容が変更されていることを確認します。

関連情報

- vSphere CSI Driver Operator

19.3.4. インストール後のパーミッションの縮小

passthrough モードを使用する場合、各コンポーネントには他のすべてのコンポーネントによって使用されるパーミッションと同じパーミッションがあります。インストール後にパーミッションを縮小しない場合、すべてのコンポーネントにはインストーラーの実行に必要な幅広いパーミッションが付与されます。

インストール後に、認証情報のパーミッションを、クラスターの実行に必要なパーミッションに制限できます。これは、使用している OpenShift Container Platform バージョンのリリースイメージの **CredentialsRequest** CR で定義されます。

AWS、Azure、または GCP に必要な **CredentialsRequest** CR を見つけ、CCO が使用するパーミッションを変更する方法については、AWS、Azure、または GCP の **IAM の手動作成**について参照してください。

19.3.5. 関連情報

- AWS の IAM の手動作成
- Azure の IAM の手動作成
- GCP の IAM の手動作成

19.4. 手動モードの使用

手動モードは、Alibaba Cloud、Amazon Web Services (AWS)、Microsoft Azure、IBM Cloud、および Google Cloud Platform (GCP) でサポートされています。

手動モードでは、ユーザーは Cloud Credential Operator (CCO) の代わりにクラウド認証情報を管理します。このモードを使用するには、実行またはインストールしている OpenShift Container Platform バージョンのリリースイメージの **CredentialsRequest** CR を検査し、基礎となるクラウドプロバイダーで対応する認証情報を作成し、クラスターのクラウドプロバイダーのすべての **CredentialsRequest** CR に対応するために Kubernetes Secret を正しい namespace に作成する必要があります。

手動モードを使用すると、クラスターに管理者レベルの認証情報を保存する必要なく、各クラスターコンポーネントに必要なパーミッションのみを指定できます。このモードでは、AWS パブリック IAM エンドポイントへの接続も必要ありません。ただし、各アップグレードについて、パーミッションを新規リリースイメージを使用して手動で調整する必要があります。

手動モードを使用するようにクラウドプロバイダーを設定する方法については、クラウドプロバイダーの手動認証情報管理オプションを参照してください。

- Alibaba Cloud の RAM リソースを手動で作成する
- AWS の IAM の手動作成
- Azure の IAM の手動作成
- GCP の IAM の手動作成
- IBM Cloud 用の IAM の設定
- Nutanix の IAM の設定

19.4.1. クラスター外で作成および管理されるクラウド認証情報を使用する手動モード

手動モードを使用する AWS または GCP クラスターは、AWS Security Token Service (STS) または GCP Workload Identity を使用して、クラスターの外部からクラウド認証情報を作成および管理するように設定されている場合があります。この設定では、CCO は異なるコンポーネントに一時的な認証情報を使用します。

詳細は、Amazon Web Services Security Token Service での手動モードの使用 または GCP ワークロードアイデンティティーでの手動モードの使用 を参照してください。

19.4.2. 手動で維持された認証情報によるクラウドプロバイダーリソースの更新

手動でメンテナンスされる認証情報でクラスターをアップグレードする前に、アップグレードするリリースイメージ用に認証情報を新規作成する必要があります。また、既存の認証情報に必要なアクセス許可を確認し、それらのコンポーネントの新しいリリースでの新しいアクセス許可要件に対応する必要があります。

手順

1. 新規リリースの **CredentialsRequest** カスタムリソースを抽出して検査します。
   クラウドプロバイダーのインストールコンテンツの IAM の手動作成についてのセクションでは、クラウドに必要な認証情報を取得し、使用する方法について説明します。
2. クラスターで手動でメンテナンスされる認証情報を更新します。
   - 新規リリースイメージによって追加される **CredentialsRequest** カスタムリソースの新規のシークレットを作成します。
   - シークレットに保存される既存の認証情報の **CredentialsRequest** カスタムリソースにパーミッション要件を変更した場合は、必要に応じてパーミッションを更新します。
3. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

   AWS 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_30_machine-api-operator_00_credentials-request.yaml (1)
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml (2)
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml (3)
```

```
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml (4)
0000_50_cluster-network-operator_02-cncc-credentials.yaml (5)
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml (6)
```

1. Machine API Operator CR が必要です。
2. Cloud Credential Operator CR が必要です。
3. Image Registry Operator CR が必要です。
4. Ingress Operator CR が必要です。
5. Network Operator CR が必要です。
6. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

GPC 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml (1)
0000_30_machine-api-operator_00_credentials-request.yaml (2)
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml (3)
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml (4)
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml (5)
0000_50_cluster-network-operator_02-cncc-credentials.yaml (6)
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml (7)
```

1. Cloud Controller Manager Operator CR が必要です。
2. Machine API Operator CR が必要です。
3. Cloud Credential Operator CR が必要です。
4. Image Registry Operator CR が必要です。
5. Ingress Operator CR が必要です。
6. Network Operator CR が必要です。
7. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

次のステップ

- **upgradeable-to** アノテーションを更新して、クラスターをアップグレードする準備ができていることを示します。

19.4.2.1. クラスターがアップグレードの準備ができていることを示す

手動で維持された認証情報をを含むクラスターの Cloud Credential Operator (CCO) の **upgradable** ステータスはデフォルトで **false** となります。

前提条件

- アップグレード先のリリースイメージについて、手動で、または Cloud Credential Operator ユーティリティー (**ccoctl**) を使用して、新しい認証情報を処理しました。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. **cluster-admin** ロールを持つユーザーとしてクラスターの **oc** にログインします。
2. 次のコマンドを実行して **CloudCredential** リソースを編集し、**metadata** フィールド内に **upgradeable-to** アノテーションを追加します。

```
$ oc edit cloudcredential cluster
```

追加するテキスト

```
...
  metadata:
    annotations:
      cloudcredential.openshift.io/upgradeable-to: <version_number>
...
```

<version_number> はアップグレード先のバージョンで、形式は **xyz** です。たとえば、OpenShift Container Platform 4.12.2 には **4.12.2** を使用します。

アノテーションを追加してから、upgradeable のステータスが変更されるまで、数分かかる場合があります。

検証

1. Web コンソールの **Administrator** パースペクティブで、**Administration → Cluster Settings** に移動します。
2. CCO ステータスの詳細を表示するには、**Cluster Operators** リストで **cloud-credential** をクリックします。
   - **Conditions** セクションの **Upgradeable** ステータスが **False** の場合に、**upgradeable-to** アノテーションに間違いがないことを確認します。
3. **Conditions** セクションの **Upgradeable** ステータスが **True** の場合、OpenShift Container Platform のアップグレードを開始します。

19.4.3. 関連情報

- Alibaba Cloud の RAM リソースを手動で作成する
- AWS の IAM の手動作成
- Amazon Web Services Security Token Service での手動モードの使用
- Azure の IAM の手動作成
- GCP の IAM の手動作成

- GCP ワークロード ID で手動モードを使用する
- IBM Cloud 用の IAM の設定
- Nutanix の IAM の設定

19.5. AMAZON WEB SERVICES SECURITY TOKEN SERVICE での手動モードの使用

STS を使用した手動モードは Amazon Web Services (AWS) でサポートされます。

注記

このクレデンシャルストラテジーは、新しい OpenShift Container Platform クラスターでのみサポートされており、インストール中に設定する必要があります。この機能を使用するために、既存のクラスターが別のクレデンシャルストラテジーを使用するように再設定することはできません。

19.5.1. AWS Security Token Service での手動モード

STS での手動モードでは、個別の OpenShift Container Platform クラスターコンポーネントは AWS Security Token Service (STS) を使用して、短期的かつ権限が制限されたセキュリティー認証情報を提供する IAM ロールをコンポーネントに割り当てます。これらの認証情報は、AWS API 呼び出しを行う各コンポーネントに固有の IAM ロールに関連付けられます。

19.5.1.1. AWS Security Token Service の認証プロセス

AWS Security Token Service (STS) と **AssumeRole** API アクションを使用すると、Pod が IAM ロールポリシーで定義されたアクセスキーを取得できるようになります。

OpenShift Container Platform クラスターには、Kubernetes サービスアカウント署名サービスが組み込まれています。このサービスは、秘密鍵を使用してサービスアカウント JSON Web トークン (JWT) に署名します。サービスアカウントトークンを必要とする Pod は、Pod 仕様を通じてサービスアカウントトークンを要求します。Pod が作成されてノードに割り当てられると、そのノードがサービスアカウント署名サービスから署名付きサービスアカウントを取得し、それを Pod にマウントします。

STS を使用するクラスターでは、Kubernetes 設定シークレットに IAM ロール ID が含まれています。ワークロードは、この IAM ロール ID のアイデンティティーを引き継ぎます。ワークロードに発行された署名付きサービスアカウントトークンが AWS の設定と一致するため、AWS STS は指定された IAM ロールのアクセスキーをワークロードに付与できます。

AWS STS は、次の条件を満たすサービスアカウントトークンを含む要求に対してのみアクセスキーを付与します。

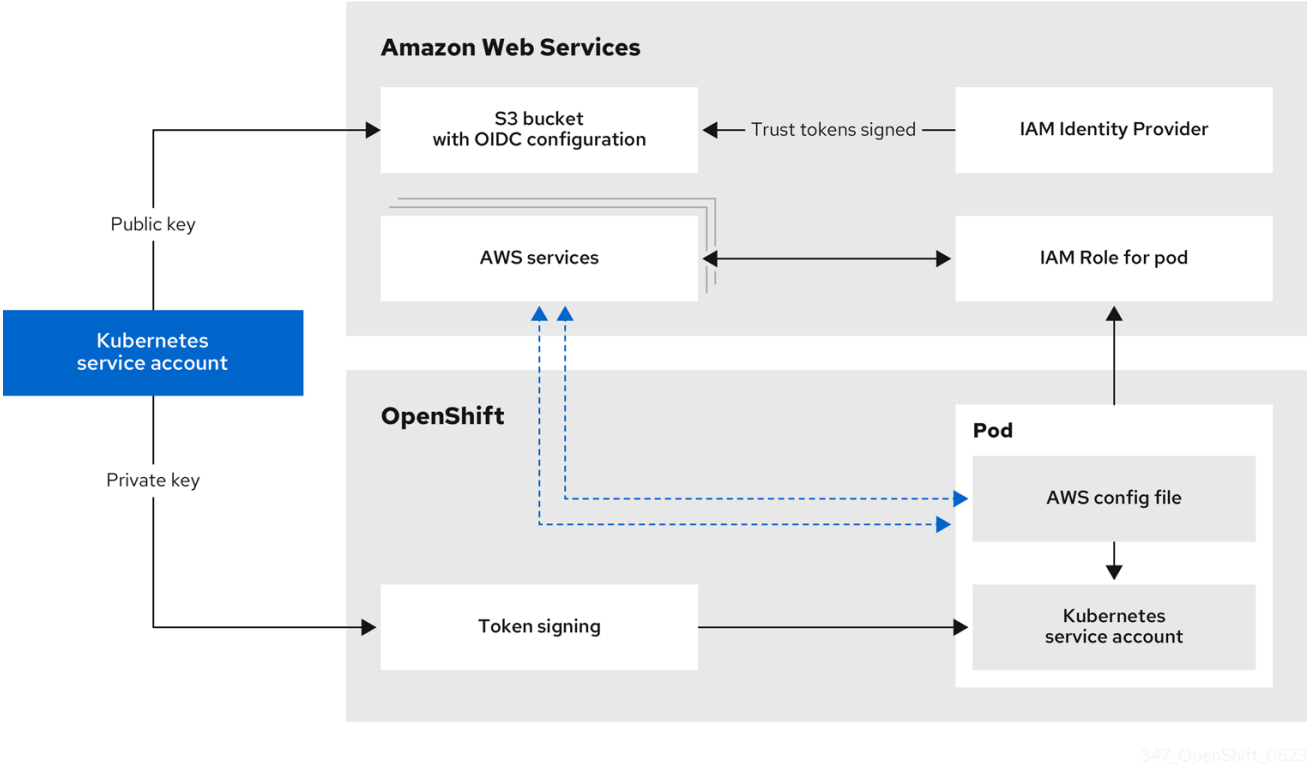
- トークンの名前と namespace が、サービスアカウントの名前と namespace と一致している。
- トークンが公開鍵と一致する鍵によって署名されている。クラスターで使用されるサービスアカウント署名鍵の公開鍵ペアは、AWS S3 バケットに保存されます。AWS STS のフェデレーションによって、サービスアカウントトークンの署名が S3 バケットに保存されている公開鍵と一致していることが検証されます。

19.5.1.2. AWS STS の認証フロー

次の図は、AWS STS を使用する場合の AWS と OpenShift Container Platform クラスター間の認証フローを示しています。

- **トークン署名** は、OpenShift Container Platform クラスター上の Kubernetes サービスアカウント署名サービスです。
- Pod 内の **Kubernetes サービスアカウント** は、署名付きサービスアカウントトークンです。

図19.2 AWS Security Token Service の認証フロー



新しい認証情報と更新された認証情報の要求は、適切に設定された AWS IAM OpenID Connect (OIDC) ID プロバイダーと AWS IAM ロールを組み合わせて使用することで自動化されます。AWS IAM によって信頼されているサービスアカウントトークンは、OpenShift Container Platform によって署名されており、Pod に展開して認証に使用できます。

19.5.1.3. AWS STS のトークン更新

Pod が使用する署名付きサービスアカウントトークンは、一定期間が経過すると期限切れになります。AWS STS を使用するクラスターの場合、この期間は 3600 秒、つまり 1 時間です。

トークンの更新は、Pod が割り当てられているノード上の kubelet により実行されます。kubelet は、トークンの有効期間の 80% を超えると、トークンのローテーションを試みます。

19.5.1.4. AWS STS の OpenID Connect の要件

OIDC 設定の暗号鍵のパブリック部分は、パブリックまたはプライベート S3 バケットに保存できます。

OIDC 仕様では HTTPS を使用する必要があります。AWS サービスは、OIDC ドキュメントを JSON Web Key Set (JWKS) 公開鍵の形式で公開するパブリックエンドポイントを必須としています。これにより、AWS サービスは、Kubernetes によって署名されたバインドされたトークンを検証し、証明書を

信頼するかどうかを判断できるようになります。そのため、どちらの S3 バケットオプションでもパブリック HTTPS エンドポイントは必要です。プライベートエンドポイントはサポートされません。

AWS STS を使用するには、AWS STS サービスのパブリック AWS バックボーンが、パブリック S3 バケット、またはパブリック CloudFront エンドポイントを備えたプライベート S3 バケットと通信できる必要があります。インストール時に、**CredentialsRequest** オブジェクトの処理に使用するバケットのタイプを選択できます。

- デフォルトでは、CCO ユーティリティー (**ccoctl**) が、OIDC 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。
- または、**ccoctl** ユーティリティーで OIDC 設定をプライベート S3 バケットに保存し、そのバケットに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーからアクセスすることもできます。

19.5.1.5. AWS コンポーネントのシークレット形式

STS で手動モードを使用すると、個別の OpenShift Container Platform コンポーネントに提供される AWS 認証情報の内容が変更されます。次のシークレット形式を比較してください。

有効期間の長い認証情報を使用した AWS シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <target-namespace> (1)
  name: <target-secret-name> (2)
data:
  aws_access_key_id: <base64-encoded-access-key-id>
  aws_secret_access_key: <base64-encoded-secret-access-key>
```

1. コンポーネントの namespace。
2. コンポーネントシークレットの名前。

STS での AWS シークレット形式

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <target-namespace> (1)
  name: <target-secret-name> (2)
stringData:
  credentials: |-
    [default]
    sts_regional_endpoints = regional
    role_name: <operator-role-name> (3)
    web_identity_token_file: <path-to-token> (4)
```

1. コンポーネントの namespace。
2. コンポーネントシークレットの名前。

3. コンポーネントの IAM ロール。
4. Pod 内のサービスアカウントトークンへのパス。通常、これは OpenShift Container Platform コンポーネントの **/var/run/secrets/openshift/serviceaccount/token** です。

19.5.2. STS での手動モードに設定された OpenShift Container Platform クラスターのインストール

STS と共に手動モードを使用して Cloud Credential Operator (CCO) を使用するように設定されるクラスターをインストールするには、以下を実行します。

1. Cloud Credential Operator ユーティリティーを設定します。
2. 必要な AWS リソースを 個別に 作成するか、1 つのコマンドを使用して 作成します。
3. OpenShift Container Platform インストーラーを実行します。
4. クラスターが有効期限の短い認証情報を使用していることを確認します。

注記

STS を使用する際にクラスターは手動モードで動作するため、必要なパーミッションでコンポーネントの新規の認証情報を作成することはできません。OpenShift Container Platform の別のマイナーバージョンにアップグレードする際に、AWS パーミッションの要件が加わることがよくあります。STS を使用するクラスターをアップグレードする前に、クラスター管理者は、AWS パーミッションが既存のコンポーネントについて使用でき、新規コンポーネントで利用可能であることを手動で確認する必要があります。

関連情報

- クラスター更新のための Cloud Credential Operator ユーティリティーの設定

19.5.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。
- **ccoctl** ユーティリティー用の AWS アカウントを作成し、次の権限で使用できるようにしました。

表19.3 必要な AWS パーミッション

パーミッションのタイプ	必須のパーミッション
iam パーミッション	○ **iam:CreateOpenIDConnectProvider** ○ **iam:CreateRole** ○ **iam:DeleteOpenIDConnectProvider** ○ **iam:DeleteRole** ○ **iam:DeleteRolePolicy** ○ **iam:GetOpenIDConnectProvider** ○ **iam:GetRole** ○ **iam:GetUser** ○ **iam:ListOpenIDConnectProviders** ○ **iam:ListRolePolicies** ○ **iam:ListRoles** ○ **iam:PutRolePolicy** ○ **iam:TagOpenIDConnectProvider** ○ **iam:TagRole**

パーミッションのタイプ	必須のパーミッション
s3 パーミッション	○ **s3:CreateBucket** ○ **s3:DeleteBucket** ○ **s3:DeleteObject** ○ **s3:GetBucketAcl** ○ **s3:GetBucketTagging** ○ **s3:GetObject** ○ **s3:GetObjectAcl** ○ **s3:GetObjectTagging** ○ **s3:ListBucket** ○ **s3:PutBucketAcl** ○ **s3:PutBucketPolicy** ○ **s3:PutBucketPublicAccessBlock** ○ **s3:PutBucketTagging** ○ **s3:PutObject** ○ **s3:PutObjectAcl** ○ **s3:PutObjectTagging**
cloudfront パーミッション	○ **cloudfront:ListCloudFrontOriginAccessIdentities** ○ **cloudfront:ListDistributions** ○ **cloudfront:ListTagsForResource**

OIDC 設定を、パブリック CloudFront ディストリビューション URL 経由で IAM アイデンティティープロバイダーがアクセスするプライベート S3 バケットに保存する予定の場合、**ccoctl** ユーティリティーを実行する AWS アカウントには次の追加パーミッションが必要です。

- **cloudfront:CreateCloudFrontOriginAccessIdentity**
- **cloudfront:CreateDistribution**
- **cloudfront:DeleteCloudFrontOriginAccessIdentity**
- **cloudfront:DeleteDistribution**
- **cloudfront:GetCloudFrontOriginAccessIdentity**
- **cloudfront:GetCloudFrontOriginAccessIdentityConfig**

- **cloudfront:GetDistribution**
- **cloudfront:TagResource**
- **cloudfront:UpdateDistribution**

注記

これらの追加のパーミッションは、**ccoctl aws create-all** コマンドで認証情報要求を処理する際の **--create-private-s3-bucket** オプションの使用をサポートします。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージを取得します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE --file="/usr/bin/ccoctl" -a ~/.pull-secret
```

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]
```

```
Available Commands:
  alibabacloud Manage credentials objects for alibaba cloud
  aws          Manage credentials objects for AWS cloud
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for IBM Cloud
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

19.5.2.2. Cloud Credential Operator ユーティリティーを使用した AWS リソースの作成

CCO ユーティリティー (**ccoctl**) を使用して、必要な AWS リソースを 個別に 作成したり、1 つのコマンドを使用して 作成したりできます。

19.5.2.2.1. AWS リソースの個別の作成

AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がある場合や、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たさない場合は、AWS リソースを個別に作成できます。たとえば、このオプションは、異なるユーザーや部門間でこれらのリソースを作成する責任を共有する組織に役に立ちます。

それ以外の場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースを自動的に作成できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

一部の **ccoctl** コマンドは AWS API 呼び出しを行い、AWS リソースを作成または変更します。**--dry-run** フラグを使用して、API 呼び出しを回避できます。このフラグを使用すると、代わりにローカルファイルシステムに JSON ファイルが作成されます。JSON ファイルを確認して変更し、AWS CLI ツールで **--cli-input-json** パラメーターを使用して適用できます。

前提条件

- **ccoctl** バイナリーを展開して準備しておく。

手順

1. クラスターの OpenID Connect プロバイダーを設定するために使用されるパブリックおよびプライベート RSA キーファイルを生成します。

```
$ ccoctl aws create-key-pair
```

出力例:

```
2021/04/13 11:01:02 Generating RSA keypair
2021/04/13 11:01:03 Writing private key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.private
2021/04/13 11:01:03 Writing public key to /<path_to_ccoctl_output_dir>/serviceaccount-signer.public
2021/04/13 11:01:03 Copying signing key for use by installer
```

serviceaccount-signer.private および **serviceaccount-signer.public** は、生成されるキーファイルです。

このコマンドは、クラスターがインストール時に必要とするプライベートキーを **/<path_to_ccoctl_output_dir>/tls/bound-service-account-signing-key.key** に作成します。

2. AWS で OpenID Connect アイデンティティープロバイダーおよび S3 バケットを作成します。

```
$ ccoctl aws create-identity-provider \
  --name=<name> \
  --region=<aws_region> \
  --public-key-file=<path_to_ccoctl_output_dir>/serviceaccount-signer.public
```

ここでは、以下のようになります。

- **<name>** は、追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
- **<aws_region>** は、クラウドリソースが作成される AWS リージョンです。
- **<path_to_ccoctl_output_dir>** は、**ccoctl aws create-key-pair** コマンドが生成したパブリックキーファイルへのパスです。

出力例:

```
2021/04/13 11:16:09 Bucket <name>-oidc created
2021/04/13 11:16:10 OpenID Connect discovery document in the S3 bucket <name>-oidc at .well-known/openid-configuration updated
2021/04/13 11:16:10 Reading public key
2021/04/13 11:16:10 JSON web key set (JWKS) in the S3 bucket <name>-oidc at keys.json updated
2021/04/13 11:16:18 Identity Provider created with ARN: arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.<aws_region>.amazonaws.com
```

openid-configuration は検出ドキュメントであり、**keys.json** は JSON Web キーセットファイルです。

このコマンドは、YAML 設定ファイルを **/<path_to_ccoctl_output_dir>/manifests/cluster-authentication-02-config.yaml** にも作成します。このファイルは、AWS IAM アイデンティティープロバイダーがトークンを信頼するように、クラスターが生成するサービスアカウントトークンの発行側の URL フィールドを設定します。

3. クラスターの各コンポーネントについて IAM ロールを作成します。

   a. OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトの一覧を抽出します。

```
$ oc adm release extract \
```

```
--from=$RELEASE_IMAGE \
--credentials-requests \
--cloud=aws \
--to=<path_to_directory_with_list_of_credentials_requests>/credrequests (1)
```

1. **credrequests** は、**CredentialsRequest** オブジェクトのリストが格納されるディレクトリーです。ディレクトリーが存在しない場合、このコマンドはディレクトリーを作成します。

b. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

AWS 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_30_cluster-api_00_credentials-request.yaml (1)
0000_30_machine-api-operator_00_credentials-request.yaml (2)
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml (3)
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml (4)
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml (5)
0000_50_cluster-network-operator_02-cncc-credentials.yaml (6)
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml (7)
```

1. **TechPreviewNoUpgrade** 機能セットを使用するクラスターの場合、Cluster API Operator CR が必要です。
2. Machine API Operator CR が必要です。
3. Cloud Credential Operator CR が必要です。
4. Image Registry Operator CR が必要です。
5. Ingress Operator CR が必要です。
6. Network Operator CR が必要です。
7. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

c. **ccoctl** ツールを使用して、**credrequests** ディレクトリーですべての **CredentialsRequest** オブジェクトを処理します。

```
$ ccoctl aws create-iam-roles \
  --name=<name> \
  --region=<aws_region> \
  --credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \
  --identity-provider-arn=arn:aws:iam::<aws_account_id>:oidc-provider/<name>-oidc.s3.
<aws_region>.amazonaws.com
```

注記

GovCloud などの代替の IAM API エンドポイントを使用する AWS 環境では、**--region** パラメーターでリージョンを指定する必要もあります。

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

それぞれの **CredentialsRequest** オブジェクトについて、**ccoctl** は指定された OIDC アイデンティティープロバイダーに関連付けられた信頼ポリシーと、OpenShift Container Platform リリースイメージの各 **CredentialsRequest** オブジェクトに定義されるパーミッションポリシーを使用して IAM ロールを作成します。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例:

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールのリスト表示について参照してください。

19.5.2.2.2. 単一コマンドでの AWS リソースの作成

AWS リソースの変更前に **ccoctl** ツールが作成する JSON ファイルを確認する必要がない場合で、**ccoctl** ツールが AWS リソースを自動作成するために使用するプロセスが組織の要件を満たす場合は、**ccoctl aws create-all** コマンドを使用して AWS リソースの作成を自動化できます。

それ以外の場合は、AWS リソースを個別に作成できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 以下のコマンドを実行して、**$RELEASE_IMAGE** 変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --cloud=aws \
  --to=<path_to_directory_with_list_of_credentials_requests>/credrequests (1)
```

(1) **credrequests** は、**CredentialsRequest** オブジェクトのリストが格納されるディレクトリーです。ディレクトリーが存在しない場合、このコマンドはディレクトリーを作成します。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

AWS 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_30_cluster-api_00_credentials-request.yaml (1)
0000_30_machine-api-operator_00_credentials-request.yaml (2)
0000_50_cloud-credential-operator_05-iam-ro-credentialsrequest.yaml (3)
0000_50_cluster-image-registry-operator_01-registry-credentials-request.yaml (4)
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml (5)
0000_50_cluster-network-operator_02-cncc-credentials.yaml (6)
0000_50_cluster-storage-operator_03_credentials_request_aws.yaml (7)
```

(1) **TechPreviewNoUpgrade** 機能セットを使用するクラスターの場合、Cluster API Operator CR が必要です。

(2) Machine API Operator CR が必要です。

(3) Cloud Credential Operator CR が必要です。

(4) Image Registry Operator CR が必要です。

(5) Ingress Operator CR が必要です。

(6) Network Operator CR が必要です。

7. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

4. **ccoctl** ツールを使用して、**credrequests** ディレクトリーですべての **CredentialsRequest** オブジェクトを処理します。

```
$ ccoctl aws create-all \
  --name=<name> \ (1)
  --region=<aws_region> \ (2)
  --credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests \ (3)
  --output-dir=<path_to_ccoctl_output_dir> \ (4)
  --create-private-s3-bucket (5)
```

1. 追跡用に作成されたクラウドリソースにタグを付けるために使用される名前です。
2. クラウドリソースが作成される AWS リージョンです。
3. コンポーネント **CredentialsRequest** オブジェクトのファイルを含むディレクトリーを指定します。
4. オプション: **ccoctl** ユーティリティーがオブジェクトを作成するディレクトリーを指定します。デフォルトでは、ユーティリティーは、コマンドが実行されるディレクトリーにオブジェクトを作成します。
5. オプション: デフォルトでは、**ccoctl** ユーティリティーは OpenID Connect (OIDC) 設定ファイルをパブリック S3 バケットに保存し、S3 URL をパブリック OIDC エンドポイントとして使用します。代わりに、パブリック CloudFront 配布 URL を介して IAM ID プロバイダーによってアクセスされるプライベート S3 バケットに OIDC 設定を保存するには、**--create-private-s3-bucket** パラメーターを使用します。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例:

```
cluster-authentication-02-config.yaml
openshift-cloud-credential-operator-cloud-credential-operator-iam-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capa-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-ebs-cloud-credentials-credentials.yaml
```

```
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-aws-cloud-credentials-credentials.yaml
```

AWS にクエリーを実行すると、IAM ロールが作成されていることを確認できます。詳細は AWS ドキュメントの IAM ロールのリスト表示について参照してください。

19.5.2.3. インストーラーの実行

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform リリースイメージを取得します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、**install-config.yaml** ファイルを作成します。

```
$ openshift-install create install-config --dir <installation_directory>
```

ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

サンプル install-config.yaml 設定ファイル

```
apiVersion: v1
baseDomain: cluster1.example.com
credentialsMode: Manual (1)
compute:
- architecture: amd64
  hyperthreading: Enabled
```

(1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

3. 必要な OpenShift Container Platform インストールマニフェストを作成します。

```
$ openshift-install create manifests
```

4. **ccoctl** によって生成されたマニフェストを、インストールプログラムが作成した manifests ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. **ccoctl** が **tls** ディレクトリーに生成したプライベートキーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6. OpenShift Container Platform インストーラーを実行します。

```
$ ./openshift-install create cluster
```

19.5.2.4. インストールの検証

1. OpenShift Container Platform クラスターに接続します。
2. クラスターに **root** 認証情報がないことを確認します。

```
$ oc get secrets -n kube-system aws-creds
```

出力は以下のようになります。

```
Error from server (NotFound): secrets "aws-creds" not found
```

3. コンポーネントが、CCO によって作成される認証情報を使用するのではなく、シークレットマニフェストで指定された IAM ロールを持つことを確認します。

Image Registry Operator を使用したコマンドの例

```
$ oc get secrets -n openshift-image-registry installer-cloud-credentials -o json | jq -r
.data.credentials | base64 --decode
```

出力には、コンポーネントによって使用されるロールおよび Web アイデンティティートークンが表示され、以下のように表示されるはずです。

Image Registry Operator を使用した出力例

```
[default]
role_arn = arn:aws:iam::123456789:role/openshift-image-registry-installer-cloud-credentials
web_identity_token_file = /var/run/secrets/openshift/serviceaccount/token
```

19.5.3. 関連情報

- 手動で維持された認証情報でクラスターを更新する準備

19.6. GCP ワークロード ID で手動モードを使用する

Google Cloud Platform (GCP) では、GCP ワークロード ID を使用した手動モードがサポートされています。

注記

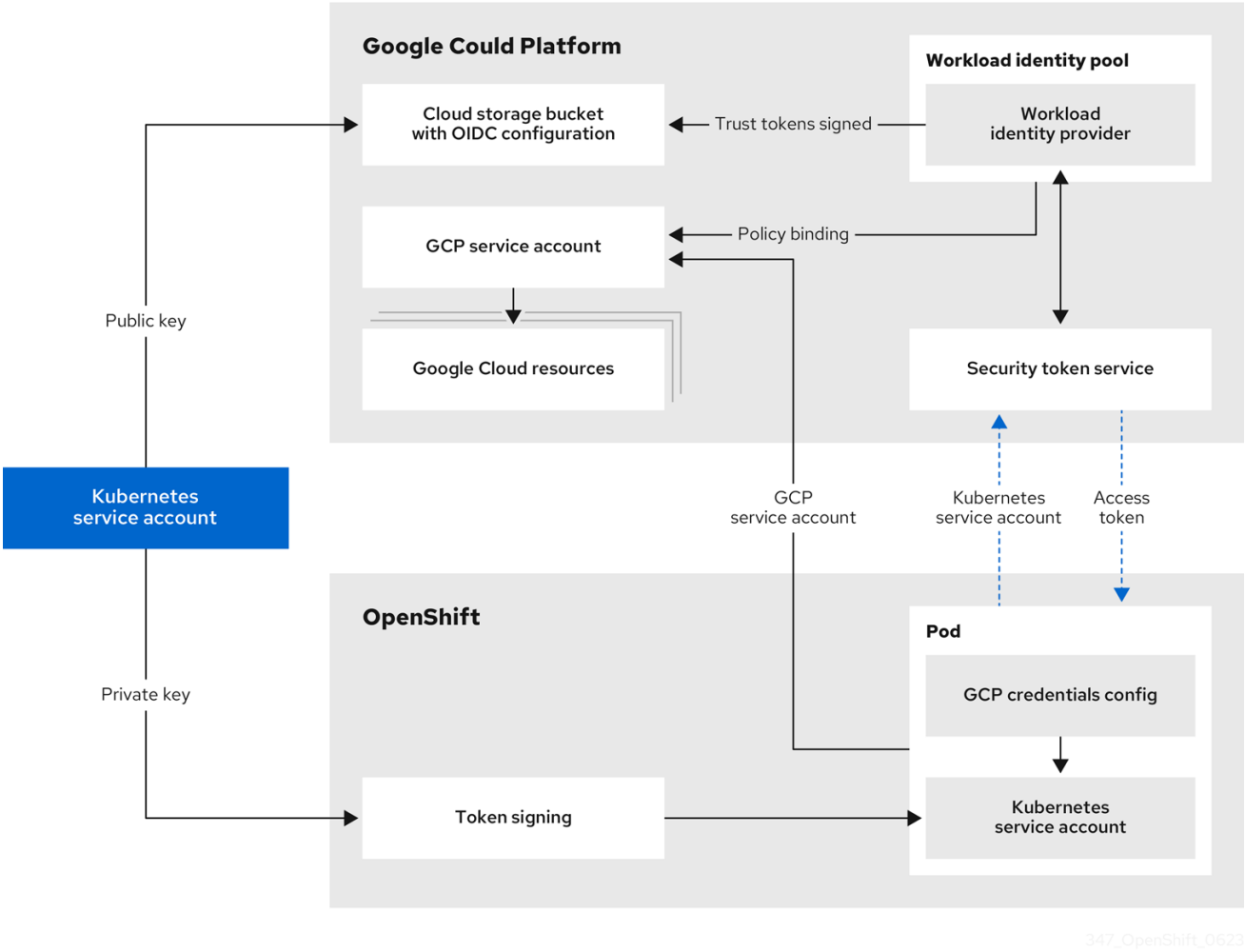
このクレデンシャルストラテジーは、新しい OpenShift Container Platform クラスターでのみサポートされており、インストール中に設定する必要があります。この機能を使用するために、既存のクラスターが別のクレデンシャルストラテジーを使用するように再設定することはできません。

19.6.1. GCP Workload ID の手動モードについて

GCP ワークロード ID を使用する手動モードでは、個々の OpenShift Container Platform クラスターコンポーネントは、短期間の限定された特権のクレデンシャルを使用して IAM サービスアカウントを偽装できます。

新しい認証情報と更新された認証情報の要求は、適切に設定された OpenID Connect (OIDC) ID プロバイダーと IAM サービスアカウントを組み合わせて使用することで自動化されます。GCP によって信頼されているサービスアカウントトークンは、OpenShift Container Platform によって署名されており、Pod に展開して認証に使用できます。トークンは 1 時間後に更新されます。

図19.3 Workload ID の認証フロー



GCP Workload Identity で手動モードを使用すると、個々の OpenShift Container Platform コンポーネントに提供される GCP クレデンシャルのコンテンツが変更されます。

GCP シークレットフォーマット

```
apiVersion: v1
kind: Secret
metadata:
  namespace: <target_namespace> (1)
  name: <target_secret_name> (2)
data:
  service_account.json: <service_account> (3)
```

1. コンポーネントの namespace。

2. コンポーネントシークレットの名前。
3. Base64 でエンコードされたサービスアカウント。

長期間有効なクレデンシャルを使用した Base64 でエンコードされた service_account.json ファイルのコンテンツ

```
{
  "type": "service_account", (1)
  "project_id": "<project_id>",
  "private_key_id": "<private_key_id>",
  "private_key": "<private_key>", (2)
  "client_email": "<client_email_address>",
  "client_id": "<client_id>",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url":
"https://www.googleapis.com/robot/v1/metadata/x509/<client_email_address>"
}
```

1. クレデンシャルの種類は **service_account** です。
2. GCP への認証に使用される秘密 RSA キー。このキーは安全に保管する必要があり、回転させないでください。

GCP Workload Identity を使用した Base64 でエンコードされた service_account.json ファイルのコンテンツ

```
{
  "type": "external_account", (1)
  "audience": "//iam.googleapis.com/projects/123456789/locations/global/workloadIdentityPools/test-
pool/providers/test-provider", (2)
  "subject_token_type": "urn:ietf:params:oauth:token-type:jwt",
  "token_url": "https://sts.googleapis.com/v1/token",
  "service_account_impersonation_url": "https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<client_email_address>:generateAccessToken", (3)
  "credential_source": {
    "file": "<path_to_token>", (4)
    "format": {
      "type": "text"
    }
  }
}
```

1. クレデンシャルの種類は**external_account**です。
2. ターゲットオーディエンスは GCP ワークロード ID プロバイダーです。
3. これらのクレデンシャルで偽装できるサービスアカウントのリソース URL。
4. Pod 内のサービスアカウントトークンへのパス。通常、これは OpenShift Container Platform コンポーネントの **/var/run/secrets/openshift/serviceaccount/token** です。

19.6.2. GCP Workload Identity ID を使用して手動モード用に設定された OpenShift Container Platform クラスターのインストール

Cloud Credential Operator (CCO) を手動モードで GCP Workload Identity を使用して使用するように設定されたクラスターをインストールするには、次の手順を実行します。

1. Cloud Credential Operator ユーティリティーを設定します。
2. 必要な GCP リソースを作成します。
3. OpenShift Container Platform インストーラーを実行します。
4. クラスターが有効期限の短い認証情報を使用していることを確認します。

注記

GCP Workload Identity を使用している場合、クラスターは手動モードで動作しているため、必要な権限を持つコンポーネントの新しい認証情報を作成できません。OpenShift Container Platform の別のマイナーバージョンにアップグレードする際に、GCP パーミッションの要件が加わることがよくあります。GCP Workload Identity を使用しているクラスターをアップグレードする前に、クラスター管理者は、GCP 権限が既存のコンポーネントに対して十分であり、新しいコンポーネントで利用できることを手動で確認する必要があります。

関連情報

- クラスター更新のための Cloud Credential Operator ユーティリティーの設定

19.6.2.1. Cloud Credential Operator ユーティリティーの設定

Cloud Credential Operator (CCO) が手動モードで動作しているときにクラスターの外部からクラウドクレデンシャルを作成および管理するには、CCO ユーティリティー (**ccoctl**) バイナリーを抽出して準備します。

注記

ccoctl ユーティリティーは、Linux 環境で実行する必要がある Linux バイナリーです。

前提条件

- クラスター管理者のアクセスを持つ OpenShift Container Platform アカウントを使用できる。
- OpenShift CLI (**oc**) がインストールされている。

手順

1. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージを取得します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから CCO コンテナーイメージを取得します。

```
$ CCO_IMAGE=$(oc adm release info --image-for='cloud-credential-operator'
$RELEASE_IMAGE -a ~/.pull-secret)
```

注記

$RELEASE_IMAGE のアーキテクチャーが、**ccoctl**ツールを使用する環境のアーキテクチャーと一致していることを確認してください。

3. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージ内の CCO コンテナーイメージから **ccoctl** バイナリーを抽出します。

```
$ oc image extract $CCO_IMAGE --file="/usr/bin/ccoctl" -a ~/.pull-secret
```

4. 次のコマンドを実行して、権限を変更して **ccoctl** を実行可能にします。

```
$ chmod 775 ccoctl
```

検証

- **ccoctl** を使用する準備ができていることを確認するには、次のコマンドを実行してヘルプファイルを表示します。

```
$ ccoctl --help
```

ccoctl --help の出力

```
OpenShift credentials provisioning tool

Usage:
  ccoctl [command]

Available Commands:
  alibabacloud Manage credentials objects for alibaba cloud
  aws          Manage credentials objects for AWS cloud
  gcp          Manage credentials objects for Google cloud
  help         Help about any command
  ibmcloud     Manage credentials objects for IBM Cloud
  nutanix      Manage credentials objects for Nutanix

Flags:
  -h, --help   help for ccoctl

Use "ccoctl [command] --help" for more information about a command.
```

19.6.2.2. Cloud Credential Operator ユーティリティーを使用した GCP リソースの作成

ccoctl gcp create-all コマンドを使用して、GCP リソースの作成を自動化できます。

注記

デフォルトで、**ccoctl** はコマンドが実行されるディレクトリーにオブジェクトを作成します。オブジェクトを別のディレクトリーに作成するには、**--output-dir** フラグを使用します。この手順では、**<path_to_ccoctl_output_dir>** を使用してこの場所を参照します。

前提条件

以下が必要になります。

- **ccoctl** バイナリーを抽出して準備している。

手順

1. 以下のコマンドを実行して、**$RELEASE_IMAGE** 変数を設定します。

```
$ RELEASE_IMAGE=$(./openshift-install version | awk '/release image/ {print $3}')
```

2. 以下のコマンドを実行して、OpenShift Container Platform リリースイメージから **CredentialsRequest** オブジェクトのリストを抽出します。

```
$ oc adm release extract \
  --from=$RELEASE_IMAGE \
  --credentials-requests \
  --cloud=gcp \
  --to=<path_to_directory_with_list_of_credentials_requests>/credrequests (1)
```

(1) **credrequests** は、**CredentialsRequest** オブジェクトのリストが格納されるディレクトリーです。ディレクトリーが存在しない場合、このコマンドはディレクトリーを作成します。

注記

このコマンドの実行には少し時間がかかる場合があります。

3. クラスターでクラスター機能を使用して 1 つ以上のオプションコンポーネントを無効にする場合は、無効なコンポーネントの **CredentialsRequest** カスタムリソースを削除します。

GPC 上の OpenShift Container Platform 4.12 の credrequests ディレクトリーの内容の例

```
0000_26_cloud-controller-manager-operator_16_credentialsrequest-gcp.yaml (1)
0000_30_cluster-api_00_credentials-request.yaml (2)
0000_30_machine-api-operator_00_credentials-request.yaml (3)
0000_50_cloud-credential-operator_05-gcp-ro-credentialsrequest.yaml (4)
0000_50_cluster-image-registry-operator_01-registry-credentials-request-gcs.yaml (5)
0000_50_cluster-ingress-operator_00-ingress-credentials-request.yaml (6)
0000_50_cluster-network-operator_02-cncc-credentials.yaml (7)
0000_50_cluster-storage-operator_03_credentials_request_gcp.yaml (8)
```

1. Cloud Controller Manager Operator CR が必要です。
2. **TechPreviewNoUpgrade** 機能セットを使用するクラスターの場合、Cluster API Operator CR が必要です。
3. Machine API Operator CR が必要です。
4. Cloud Credential Operator CR が必要です。
5. Image Registry Operator CR が必要です。
6. Ingress Operator CR が必要です。
7. Network Operator CR が必要です。
8. Storage Operator CR はオプションのコンポーネントであり、クラスターで無効になっている場合があります。

4. **ccoctl** ツールを使用して、**credrequests** ディレクトリーですべての **CredentialsRequest** オブジェクトを処理します。

```
$ ccoctl gcp create-all \
  --name=<name> \
  --region=<gcp_region> \
  --project=<gcp_project_id> \
  --credentials-requests-dir=
<path_to_directory_with_list_of_credentials_requests>/credrequests
```

ここでは、以下のようになります。

- **<name>** は、トラッキングに使用される、作成されたすべての GCP リソースのユーザー定義名です。
- **<gcp_region>** は、クラウドリソースが作成される GCP リージョンです。
- **<gcp_project_id>** は、クラウドリソースが作成される GCP プロジェクト ID です。
- **<path_to_directory_with_list_of_credentials_requests>/credrequests** は、GCP サービスアカウントを作成するための**Credentials Request**マニフェストのファイルを含むディレクトリーです。

注記

クラスターで **TechPreviewNoUpgrade** 機能セットによって有効化されたテクノロジープレビュー機能を使用している場合は、**--enable-tech-preview** パラメーターを含める必要があります。

検証

- OpenShift Container Platform シークレットが作成されることを確認するには、**<path_to_ccoctl_output_dir>/manifests** ディレクトリーのファイルを一覧表示します。

```
$ ls <path_to_ccoctl_output_dir>/manifests
```

出力例:

```
cluster-authentication-02-config.yaml
openshift-cloud-controller-manager-gcp-ccm-cloud-credentials-credentials.yaml
openshift-cloud-credential-operator-cloud-credential-operator-gcp-ro-creds-credentials.yaml
openshift-cloud-network-config-controller-cloud-credentials-credentials.yaml
openshift-cluster-api-capg-manager-bootstrap-credentials-credentials.yaml
openshift-cluster-csi-drivers-gcp-pd-cloud-credentials-credentials.yaml
openshift-image-registry-installer-cloud-credentials-credentials.yaml
openshift-ingress-operator-cloud-credentials-credentials.yaml
openshift-machine-api-gcp-cloud-credentials-credentials.yaml
```

GCP にクエリーを実行すると、IAM サービスアカウントが作成されていることを確認できます。詳細については、IAM サービスアカウントのリスト表示に関する GCP のドキュメントを参照してください。

19.6.2.3. インストーラーの実行

前提条件

- クラスターをホストするクラウドプラットフォームでアカウントを設定します。
- OpenShift Container Platform リリースイメージを取得します。

手順

1. インストールプログラムが含まれるディレクトリーに切り替え、**install-config.yaml** ファイルを作成します。

   ```
   $ openshift-install create install-config --dir <installation_directory>
   ```

   ここで、**<installation_directory>** は、インストールプログラムがファイルを作成するディレクトリーに置き換えます。

2. **install-config.yaml** 設定ファイルを編集し、**credentialsMode** パラメーターが **Manual** に設定されるようにします。

   サンプル install-config.yaml 設定ファイル

   ```
   apiVersion: v1
   baseDomain: cluster1.example.com
   credentialsMode: Manual (1)
   compute:
   - architecture: amd64
     hyperthreading: Enabled
   ```

   (1) この行は、**credentialsMode** パラメーターを **Manual** に設定するために追加されます。

3. 必要な OpenShift Container Platform インストールマニフェストを作成します。

   ```
   $ openshift-install create manifests
   ```

4. **ccoctl** によって生成されたマニフェストを、インストールプログラムが作成した manifests ディレクトリーにコピーします。

```
$ cp /<path_to_ccoctl_output_dir>/manifests/* ./manifests/
```

5. **ccoctl** が **tls** ディレクトリーに生成したプライベートキーをインストールディレクトリーにコピーします。

```
$ cp -a /<path_to_ccoctl_output_dir>/tls .
```

6. OpenShift Container Platform インストーラーを実行します。

```
$ ./openshift-install create cluster
```

19.6.2.4. インストールの検証

1. OpenShift Container Platform クラスターに接続します。
2. クラスターに **root** 認証情報がないことを確認します。

```
$ oc get secrets -n kube-system gcp-credentials
```

出力は以下のようになります。

```
Error from server (NotFound): secrets "gcp-credentials" not found
```

3. コンポーネントが、CCO によって作成される認証情報を使用するのではなく、シークレットマニフェストで指定されたサービスアカウントを持つことを確認します。

Image Registry Operator を使用したコマンドの例

```
$ oc get secrets -n openshift-image-registry installer-cloud-credentials -o json | jq -r
'.data."service_account.json"' | base64 -d
```

出力には、コンポーネントによって使用されるロールおよび Web アイデンティティートークンが表示され、以下のように表示されるはずです。

Image Registry Operator を使用した出力例

```
{
  "type": "external_account", (1)
  "audience":
"//iam.googleapis.com/projects/123456789/locations/global/workloadIdentityPools/test-
pool/providers/test-provider",
  "subject_token_type": "urn:ietf:params:oauth:token-type:jwt",
  "token_url": "https://sts.googleapis.com/v1/token",
  "service_account_impersonation_url": "https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<client-email-address>:generateAccessToken", (2)
  "credential_source": {
    "file": "/var/run/secrets/openshift/serviceaccount/token",
    "format": {
      "type": "text"
    }
  }
}
```

1. クレデンシャルの種類は**external_account**です。
2. Image Registry Operator が使用するサービスアカウントのリソース URL。

19.6.3. 関連情報

- 手動で維持された認証情報でクラスターを更新する準備