



OpenShift Container Platform 4.2

認証

ユーザー認証、暗号化、およびユーザーとサービスのアクセス制御の設定

OpenShift Container Platform 4.2 認証

ユーザー認証、暗号化、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 © 2020 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 4.2 でアイデンティティプロバイダーを定義する方法を説明します。また、暗号化およびロールベースのアクセス制御を使ってクラスタのセキュリティを保護する方法についても説明します。

目次

第1章 認証について	4
1.1. ユーザー	4
1.2. グループ	4
1.3. API 認証	5
第2章 内部 OAUTH サーバーの設定	8
2.1. OPENSIFT CONTAINER PLATFORM OAUTH サーバー	8
2.2. OAUTH トークン要求フローおよび応答	8
2.3. 内部 OAUTH サーバーのオプション	9
2.4. 内部 OAUTH サーバーのトークン期間の設定	9
2.5. 追加の OAUTH クライアントの登録	10
2.6. OAUTH サーバーメタデータ	11
2.7. OAUTH API イベントのトラブルシューティング	12
第3章 アイデンティティプロバイダー設定について	14
3.1. OPENSIFT CONTAINER PLATFORM のアイデンティティプロバイダーについて	14
3.2. サポートされるアイデンティティプロバイダー	14
3.3. KUBEADMIN ユーザーの削除	15
3.4. アイデンティティプロバイダーパラメーター	15
3.5. アイデンティティプロバイダー CR のサンプル	16
第4章 アイデンティティプロバイダーの設定	18
4.1. HTTPASSWD アイデンティティプロバイダーの設定	18
4.2. KEYSTONE アイデンティティプロバイダーの設定	22
4.3. LDAP アイデンティティプロバイダーの設定	25
4.4. BASIC 認証アイデンティティプロバイダーの設定	29
4.5. 要求ヘッダーアイデンティティプロバイダーの設定	35
4.6. GITHUB または GITHUB ENTERPRISE アイデンティティプロバイダーの設定	43
4.7. GITLAB アイデンティティプロバイダーの設定	47
4.8. GOOGLE アイデンティティプロバイダーの設定	49
4.9. OPENID CONNECT アイデンティティプロバイダーの設定	52
第5章 証明書の設定	58
5.1. デフォルトの INGRESS 証明書の置き換え	58
5.2. API サーバー証明書の追加	59
5.3. サービス提供証明書のシークレットによるサービストラフィックのセキュリティ保護	61
第6章 RBAC の使用によるパーミッションの定義および適用	65
6.1. RBAC の概要	65
6.2. プロジェクトおよび NAMESPACE	68
6.3. デフォルトプロジェクト	69
6.4. クラスターロールおよびバインディングの表示	69
6.5. ローカルのロールバインディングの表示	71
6.6. ロールのユーザーへの追加	73
6.7. ローカルロールの作成	74
6.8. クラスターロールの作成	75
6.9. ローカルロールバインディングのコマンド	75
6.10. クラスターのロールバインディングコマンド	76
6.11. クラスター管理者の作成	76
第7章 KUBEADMIN ユーザーの削除	78
7.1. KUBEADMIN ユーザー	78
7.2. KUBEADMIN ユーザーの削除	78

第8章 ユーザーエージェントの設定	79
8.1. ユーザーエージェントについて	79
8.2. ユーザーエージェントの設定	79
第9章 サービスアカウントの概要および作成	81
9.1. サービスアカウントの概要	81
9.2. サービスアカウントの作成	81
9.3. ロールをサービスアカウントに付与する例	82
第10章 アプリケーションでのサービスアカウントの使用	84
10.1. サービスアカウントの概要	84
10.2. デフォルトのサービスアカウント	84
10.3. サービスアカウントの作成	85
10.4. サービスアカウントの認証情報の外部での使用	86
第11章 サービスアカウントの OAUTH クライアントとしての使用	88
11.1. OAUTH クライアントとしてのサービスアカウント	88
第12章 スコープトークン	91
12.1. トークンのスコープについて	91
第13章 SCC (SECURITY CONTEXT CONSTRAINTS) の管理	92
13.1. SCC (SECURITY CONTEXT CONSTRAINTS) について	92
13.2. 事前に割り当てられる SCC (SECURITY CONTEXT CONSTRAINTS) 値について	97
13.3. SCC (SECURITY CONTEXT CONSTRAINTS) の例	99
13.4. SCC (SECURITY CONTEXT CONSTRAINTS) の作成	101
13.5. SCC (SECURITY CONTEXT CONSTRAINTS) へのロールベースのアクセス	102
13.6. SCC (SECURITY CONTEXT CONSTRAINTS) 参照コマンド	103
第14章 SYSTEM:ADMIN ユーザーの権限の借用	106
14.1. API の権限借用	106
14.2. SYSTEM:ADMIN ユーザーの権限の借用	106
第15章 LDAP グループの同期	107
15.1. LDAP 同期の設定について	107
15.2. LDAP 同期の実行	112
15.3. グループのプルニングジョブの実行	114
15.4. LDAP グループの同期の例	114
15.5. LDAP 同期設定の仕様	127
第16章 追加ホストから API サーバーへの JAVASCRIPT ベースのアクセスの許可	135
16.1. 追加ホストから API サーバーへの JAVASCRIPT ベースのアクセスの許可	135

第1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。

1.1. 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です。OpenShift Container Platform ユーザーオブジェクトは、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。

regular user (通常ユーザー)	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 オブジェクトで表示されます。例: joe alice
system user (システムユーザー)	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 (匿名) 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 account (サービスアカウント)	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 (匿名)** 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。

1.2. 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。

1.3. 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証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。

1.3.1. 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を判別します。

次に、そのアイデンティティ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。

1.3.1.1. 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。
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。



注記

<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、認証プロキシをこのエンドポイントの前に配置するか、または OpenShift Container Platform を、サポートするアイデンティティ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。

認証プロキシが **WWW-Authenticate** チャレンジをサポートしないか、または 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。

1.3.1.2. 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。

1.3.1.3. Prometheus の認証メトリクス

OpenShift Container Platform は認証の試行中に以下の Prometheus システムメトリクス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2章 内部 OAUTH サーバーの設定



重要

以下のオプションはマスター設定ファイルに設定されているため、それらの設定によって変更が生じます。

2.1. OPENSIFT 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を判別します。

次に、そのアイデンティティ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。

2.2. 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 認証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	音声内容	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 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 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	チャレンジの承認ができません。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)

ステータス	音声内容	クライアント応答
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。

2.3. 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。

2.3.1. 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。

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。
認証コード	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上書きできます。

2.3.2. 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戦略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、独自の付与戦略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。

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。

2.4. 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。

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

2.5. 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な場合は、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
redirectURLs:
```

```
- "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、またはこれによってプレフィックス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。

2.6. 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 スコープの値の一覧を含む 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** 値の一覧を含む 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 の値の一覧が含まれる 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一覧が含まれる 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。

2.7. 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
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
```



```

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一覧です。

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3章 アイデンティティ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プロバイダーを指定するように設定できます。

3.1. OPENSIFT CONTAINER PLATFORM のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

3.2. サポートされるアイデンティティプロバイダー

以下の種類のアイデンティティプロバイダーを設定できます。

アイデンティティプロバイダー	説明
HTPasswd	htpasswd アイデンティティプロバイダーを 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 アイデンティティプロバイダーを、ユーザーがリモートアイデンティティプロバイダーに対して検証された認証情報を使って 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と併用されます。
GitHub または GitHub Enterprise	github アイデンティティプロバイダーを、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。

アイデンティティプロバイダー	説明
GitLab	gitlab アイデンティティプロバイダーを、 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プロバイダーとして使用するよう設定します。
Google	google アイデンティティプロバイダーを、 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 アイデンティティプロバイダーと統合するよう設定します。

アイデンティティ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。

3.3. KUBEADMIN ユーザーの削除

アイデンティティ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
```

3.4. アイデンティティ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プレフィックスとして付加され、アイデンティティ名が作成されます。
mappingMethod	<p>新規アイデンティティ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。</p> <p>claim</p> <p>デフォルトの値です。アイデンティティ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にマッピングされている場合は失敗します。</p> <p>lookup</p> <p>既存のアイデンティティ、ユーザーアイデンティティマッピング、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。</p> <p>generate</p> <p>アイデンティティ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既存のアイデンティティにマッピングされている場合は、一意のユーザー名が生成されます。例: myuser2この方法は、OpenShift Container Platform のユーザー名とアイデンティティプロバイダーのユーザー名との正確な一致を必要とする外部プロセス (LDAP グループ同期など) と組み合わせて使用することはできません。</p> <p>add</p> <p>アイデンティティ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プロバイダーが複数設定されている場合に必要です。</p>



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を既存ユーザーにマッピングできます。

3.5. アイデンティティプロバイダー 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 アイデンティティプロバイダーを使用しています。

アイデンティティプロバイダー CR のサンプル

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

```
mappingMethod: claim ②  
type: HTTPasswd  
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プレフィックスとして付加され、アイデンティティ名が作成されます。
- ② このプロバイダーのアイデンティティとユーザー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4章 アイデンティティプロバイダーの設定

4.1. HTPASSWD アイデンティティプロバイダーの設定

4.1.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

HTPasswd アイデンティティプロバイダーを定義するには、以下の手順を実行する必要があります。

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。Linux および Windows用の手順が説明されます。
2. **htpasswd** ファイルを表すために OpenShift Container Platform シークレットを作成します。
3. HTPasswd アイデンティティプロバイダーリソースを定義します。
4. リソースをデフォルトの OAuth 設定に適用します。

4.1.2. Linux を使用した HTPasswd ファイルの作成

HTPasswd アイデンティティ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> <user_name> <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。

例:

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

```
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続けます。

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

4.1.3. Windows を使用した HTPasswd ファイルの作成

HTPasswd アイデンティティプロバイダーを使用するには、**htpasswd**を使用してクラスタ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> <user_name> <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。

例:

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

```
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続けます。

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

4.1.4. HTPasswd シークレットの作成

HTPasswd アイデンティティ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 OpenShift Container Platform シークレットを作成します。

```
$ oc create secret generic htpass-secret --from-file=htpasswd=</path/to/users.htpasswd> -n openshift-config
```



注記

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。

4.1.5. 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 ❶
    mappingMethod: claim ❷
    type: HTPasswd
    htpasswd:
      fileData:
        name: htpass-secret ❸
```

- ❶ このプロバイダー名は、プロバイダーのユーザー名にプレフィックスとして付加され、アイデンティティ名が作成されます。
- ❷ このプロバイダーのアイデンティティとユーザー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。
- ❸ **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。

4.1.6. アイデンティティ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
```

4.1.7. HTPasswd アイデンティティプロバイダーの更新

既存の HTPasswd アイデンティティ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シークレットを作成している。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** シークレットから取得し、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 -d > 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>
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>
Deleting password for user <username>
```

3. **htpass-secret** シークレットを、**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 -o yaml -n openshift-config | oc replace -f -
```

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. ユーザーを削除します。

```
$ oc delete user <username>
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. ユーザーのアイデンティティを削除します。

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

4.1.8. Web コンソールを使用したアイデンティティ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プロバイダー (IDP) を設定します。

前提条件

- クラスタ管理者として 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. **Global 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。

4.2. KEYSTONE アイデンティティプロバイダーの設定

keystone アイデンティティプロバイダーを、OpenShift Container Platform クラスタを 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 認証情報を使って OpenShift Container Platform にログインできます。

Keystone は、アイデンティティ、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 ユーザーのベースを Keystone ID としない方法がより安全な方法になります。Keystone ユーザーを削除し、そのユーザー名で新規の Keystone ユーザーを作成する場合、新規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。

4.2.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.2.2. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。

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

4.2.3. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.2.4. 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プレフィックスとして付加され、アイデンティティ名が作成されます。
- ② このプロバイダーのアイデンティティとユーザー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 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 シークレットへの参照。
- ⑦ クライアント証明書のキーを含む OpenShift Container Platform シークレットへの参照。 **tlsClientCert** が指定されている場合には必須になります。

4.2.5. アイデンティティ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。

- アイデンティティ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>
```

- 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
```

4.3. LDAP アイデンティティプロバイダーの設定

ldap アイデンティティプロバイダーを、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。

4.3.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.3.2. 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。

- 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で検索フィルターを生成します。
- 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つもエントリが返されない場合は、アクセスを拒否します。
- 検索で取得したエントリの 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。
- バインドが失敗した場合は、アクセスを拒否します。
- バインドが成功した場合は、アイデンティティ、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、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 はカンマ区切りの属性の一覧を許可しますが、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。

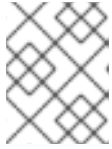
LDAP ディレクトリーの検索に認証が必要な場合は、エントリー検索の実行に使用する **bindDN** と **bindPassword** を指定します。

4.3.3. 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 シークレットを定義する必要があります。

- bindPassword が含まれる OpenShift Container Platform シークレットを定義します。

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



注記

上記のコマンドが示すように、**--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。

4.3.4. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.3.5. 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: "ldap://dap.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プレフィックスとして付加され、アイデンティティ名が作成されます。
- 2** このプロバイダーのアイデンティティとユーザー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** アイデンティティとして使用する属性の一覧です。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。一覧表示される属性のいずれにも値がない場合、認証は失敗します。定義される属性は raw データとして取得され、バイナリー値の使用を許可します。
- 4** メールアドレスとして使用する属性の一覧です。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一覧です。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一覧です。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 シークレットへの参照。 **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 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 ユーザーのアイデンティティとユーザーオブジェクトを作成する必要があります。

4.3.6. アイデンティティ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
```

4.4. BASIC 認証アイデンティティプロバイダーの設定

basic-authentication アイデンティティプロバイダーを、ユーザーがリモートアイデンティティプロバイダーに対して検証された認証情報を使って 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。

4.4.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.4.2. 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プロバイダーを使用して詳細な認証設定を実行できます。

注意

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ます、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。例:

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

- **email** キーを使用したメールアドレス。例:

```
{"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。例:

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

4.4.3. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。

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

4.4.4. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.4.5. 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 ❶
    mappingMethod: claim ❷
    type: BasicAuth
    basicAuth:
      url: https://www.example.com/remote-idp ❸
      ca: ❹
        name: ca-config-map
      tlsClientCert: ❺
        name: client-cert-secret
      tlsClientKey: ❻
        name: client-key-secret
```

- ❶ このプロバイダー名は返されるユーザー名にプレフィックスとして付加され、アイデンティティ名が作成されます。

- 2 このプロバイダーのアイデンティティとユーザー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 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 シークレットへの参照。
- 6 クライアント証明書のキーを含む OpenShift Container Platform シークレットへの参照。 **tlsClientCert** が指定されている場合には必須になります。

4.4.6. アイデンティティ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。

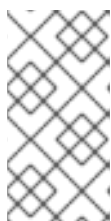
前提条件

- 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
```

4.4.7. 基本的なアイデンティティプロバイダーの 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。

/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>
```

/var/www/cgi-bin/login.cgi

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

/var/www/cgi-bin/fail.cgi

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

4.4.7.1. 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)。

4.4.8. 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

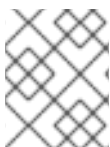
- **401** 応答は認証の失敗を示しています。
- **200** 以外のステータスまたは空でない「エラー」キーはエラーを示しています: **{"error": "Error message"}**

4.5. 要求ヘッダーアイデンティティプロバイダーの設定

request-header アイデンティティプロバイダーを、**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と併用されます。

4.5.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.5.2. 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と併用されます。



注記

さらに、コミュニティでサポートされる [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プロバイダーを使用して認証を行うには、認証プロキシ経由で **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** にプロキシ処理するプロキシ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 URL に設定してから、要求を **https://<namespace_route>/oauth/authorize** にプロキシします。

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 URL に設定し、要求を **https://<namespace_route>/oauth/authorize** にプロキシします。

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。

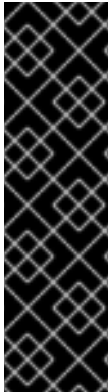
例: [https://www.example.com/auth-proxy/oauth/authorize?\\${query}](https://www.example.com/auth-proxy/oauth/authorize?${query})



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。

4.5.2.1. 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です。テクノロジープレビュー機能は Red Hat の実稼働環境でのサービスレベルアグリーメント (SLA) ではサポートされていないため、Red Hat では実稼働環境での使用を推奨していません。これらの機能は、近々発表予定の製品機能をリリースに先駆けてご提供することにより、お客様は機能性をテストし、開発プロセス中にフィードバックをお寄せいただくことができます。

Red Hat のテクノロジープレビュー機能のサポートについての詳細は、<https://access.redhat.com/support/offerings/techpreview/> を参照してください。

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ースを使用して OpenShift Container Platform に対して自動的に認証されます。

4.5.3. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.5.4. 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?${query}" ❸
  loginURL: "https://www.example.com/login-proxy/oauth/authorize?${query}" ❹
  ca: ❺
    name: ca-config-map
  clientCommonNames: ❻
  - my-auth-proxy
  headers: ❼
  - X-Remote-User
  - SSO-User
  emailHeaders: ❽
  - X-Remote-User-Email
  nameHeaders: ❾
  - X-Remote-User-Display-Name
  preferredUsernameHeaders: ❿
  - X-Remote-User-Login

```

- ❶ このプロバイダー名は要求ヘッダーのユーザー名にプレフィックスとして付加され、アイデンティティ名が作成されます。
- ❷ このプロバイダーのアイデンティティとユーザー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。
- ❸ 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` にプロキシします。 `https://<namespace_route>/oauth/authorize` にプロキシする URL は `/authorize` で終了し (末尾のスラッシュなし)、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します。 `${url}` は現在の URL に置き換えられ、エスケープされてクエリーパラメーターで保護されます。 `${query}` は現在のクエリー文字列に置き換えられます。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。
- ❹ 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。 **WWW-Authenticate** チャレンジが想定されるクライアントを認証し、それらを `https://<namespace_route>/oauth/authorize` にプロキシします。 `${url}` は現在の URL に置き換えられ、エスケープされてクエリーパラメーターで保護されます。 `${query}` は現在のクエリー文字列に置き換えられます。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。
- ❺ PEM エンコードされた証明書バンドルを含む OpenShift Container Platform ConfigMap 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プロバイダーに必要です。これは、プロキシが相互 TLS をサポートしている必要があることを意味します。

- ❻ オプション: 共通名 (`cn`) の一覧。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一覧の 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せん。

4.5.5. アイデンティティ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。

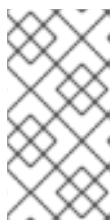
前提条件

- 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
```

4.5.6. 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プロバイダーを使用して OpenShift Container Platform の Apache 認証プロキシを設定します。

カスタムプロキシ設定

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プロバイダーを使用して Apache 認証プロキシ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へ送信できるように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?...** にプロキシする必要があります。
- **https://<namespace_route>/oauth/authorize** にプロキシされる URL のサブパスは、**https://<namespace_route>/oauth/authorize** にプロキシ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プロバイダーを使用して Apache 認証プロキシを設定します。

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 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
```

CA は、ConfigMap の **ca.crt** キーに保存する必要があります。

- このプロキシ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 設定の証明書を生成します。**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
```

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。

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。

4.6. GITHUB または GITHUB ENTERPRISE アイデンティティプロバイダーの設定

github アイデンティティプロバイダーを、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。

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。

4.6.1. OpenShift Container Platform のアイデンティティ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に存在します。アイデンティティプロバイダーを指定するには、アイデンティティプロバイダーを記述し、これをクラスタに追加するカスタムリソース (CR、Custom Resource) 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。

4.6.2. 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>
```

以下は例になります。

```
https://oauth-openshift.apps.example-openshift-cluster.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、アイデンティティプロバイダーの設定を完了するために必要です。

4.6.3. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。


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

4.6.4. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.6.5. 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 ❶
    mappingMethod: claim ❷
    type: GitHub
    github:
      ca: ❸
        name: ca-config-map
      clientID: {...} ❹
      clientSecret: ❺
        name: github-secret
      hostname: ... ❻
      organizations: ❼
        - myorganization1
        - myorganization2
      teams: ❽
        - myorganization1/team-a
        - myorganization2/team-b
```

- ❶ このプロバイダー名は GitHub の数字ユーザー ID にプレフィックスとして付加され、アイデンティティ名が作成されます。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。
- ❷ このプロバイダーのアイデンティティとユーザー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。
- ❸ 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 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-`
- 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 シークレットへの参照。
- 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ます。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 組織のオプションの一覧です。これが指定されている場合、少なくとも一覧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。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 チームのオプションの一覧です。これが指定されている場合、少なくとも一覧のいずれかのチームのメンバーである GitHub ユーザーのみがログインできます。そのチームの組織が `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。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。

4.6.6. アイデンティティ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
```

4.7. GITLAB アイデンティティプロバイダーの設定

gitlab アイデンティティプロバイダーを、[GitLab.com](https://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プロバイダーとして使用するように設定します。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、OAuth ではなく [OpenID Connect \(OIDC\)](#) を使用して接続します。

4.7.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) を作成する必要があります。



注記

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

4.7.2. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。

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

4.7.3. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.7.4. 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 にプレフィックスとして付加され、アイデンティティ名が作成されます。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とユーザー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 シークレットへの参照。
- ⑤ GitLab プロバイダーのホスト URL です。これは <https://gitlab.com/> か、または他の 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 への参照。

4.7.5. アイデンティティ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
```

4.8. GOOGLE アイデンティティプロバイダーの設定

google アイデンティティプロバイダーを、[Google の OpenID Connect 統合](#) を使用して設定します。



注記

Google をアイデンティティプロバイダーとして使用するには、**<master>/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。



警告

Google をアイデンティティ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。

4.8.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.8.2. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。

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

4.8.3. 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 にプレフィックスとして付加され、アイデンティティ名が作成されます。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とユーザー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 シークレットへの参照。
- 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。

4.8.4. アイデンティティ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
```

4.9. OPENID CONNECT アイデンティティプロバイダーの設定

oidc アイデンティティプロバイダーを、[Authorization Code Flow](#)を使用して OpenID Connect アイデンティティプロバイダーと統合するように設定します。

OpenShift Container Platform の OpenID Connect アイデンティティプロバイダーとして [Red Hat シングルサインオン](#)を設定できます。



重要

OpenShift Container Platform の認証 Operator では、設定済みの OpenID Connect アイデンティティプロバイダーが [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、値が入力されている最初の要求が使用されます。標準のアイデンティティ要求は以下の通りです。

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 アイデンティティプロバイダーを使用するには、**<master>/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。

4.9.1. OpenShift Container Platform のアイデンティティ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。アイデンティティプロバイダーを指定するには、アイデンティティ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 (CR、Custom Resource) 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。

4.9.2. シークレットの作成

アイデンティティプロバイダーは **openshift-config** namespace で OpenShift Container Platform シークレットを使用して、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。

- 以下のコマンドを使用して、OpenShift Container Platform シークレットを定義できます。

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

- 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 OpenShift Container Platform シークレットを定義できます。

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

4.9.3. ConfigMap の作成

アイデンティティプロバイダーは、**openshift-config** namespace で OpenShift Container Platform ConfigMap 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プロバイダーに必要な証明書バンドルを組み込むために使用されます。

- 以下のコマンドを使用して、認証局が含まれる OpenShift Container Platform ConfigMap を定義します。認証局は ConfigMap 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
```

4.9.4. 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。

OpenID Connect CR のサンプル

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
  - name: oidcidp ❶
    mappingMethod: claim ❷
    type: OpenID
    openID:
      clientID: ... ❸
      clientSecret: ❹
        name: idp-secret
      claims: ❺
        preferredUsername:
          - preferred_username
        name:
          - name
        email:
          - email
      issuer: https://www.idp-issuer.com ❻
```

- ❶ このプロバイダー名はアイデンティティ要求の値にプレフィックスとして付加され、アイデンティティ名が作成されます。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。
- ❷ このプロバイダーのアイデンティティとユーザー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。
- ❸ 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。
- ❹ クライアントシークレットを含む OpenShift Container Platform シークレットへの参照。
- ❺ アイデンティティとして使用する要求の一覧です。空でない最初の要求が使用されます。1つ以上の要求が必要になります。一覧表示される要求のいずれにも値がないと、認証は失敗します。たとえば、これは、ユーザーのアイデンティティとして、返される **id_token** の **sub** 要求の値を使用します。
- ❻ 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: ❶
        name: ca-config-map
      extraScopes: ❷
        - email
        - profile
      extraAuthorizeParameters: ❸
        include_granted_scopes: "true"
      claims:
        preferredUsername: ❹
          - preferred_username
          - email
        name: ❺
          - nickname
          - given_name
          - name
        email: ❻
          - custom_email_claim
          - email
      issuer: https://www.idp-issuer.com

```

- ❶ 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 への参照。
- ❷ 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オプションの一覧です。
- ❸ 認可トークン要求に追加する追加パラメーターのオプションのマップです。
- ❹ このアイデンティティ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一覧です。空でない最初の要求が使用されます。
- ❺ 表示名として使用する要求の一覧です。空でない最初の要求が使用されます。
- ❻ メールアドレスとして使用する要求の一覧です。空でない最初の要求が使用されます。

4.9.5. アイデンティティ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
```

4.9.6. Web コンソールを使用したアイデンティティ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。

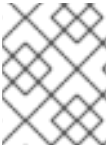
前提条件

- 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. **Global 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5章 証明書の設定

5.1. デフォルトの INGRESS 証明書の置き換え

5.1.1. デフォルトの Ingress 証明書について

デフォルトで、OpenShift Container Platform は Ingress Operator を使用して内部 CA を作成し、**.apps** サブドメインの下にあるアプリケーションに有効なワイルドカード証明書を発行します。Web コンソールと CLI のどちらもこの証明書を使用します。

内部インフラストラクチャー CA 証明書は自己署名型です。一部のセキュリティーまたは PKI チームにとってこのプロセスは適切とみなされない可能性があります。ここで想定されるリスクは最小限度のもので、これらの証明書を暗黙的に信頼するクライアントがクラスター内の他のコンポーネントになります。デフォルトのワイルドカード証明書を、コンテナユーザー空間で提供される CA バンドルにすでに含まれているパブリック CA に置き換えることで、外部クライアントは **.apps** サブドメインで実行されるアプリケーションに安全に接続できます。

5.1.2. デフォルトの Ingress 証明書の置き換え

.apps サブドメインにあるすべてのアプリケーションのデフォルトの Ingress 証明書を置き換えることができます。証明書を置き換えた後に、Web コンソールや CLI を含むすべてのアプリケーションには、指定された証明書で提供される暗号化が設定されます。

前提条件

- ワイルドカード証明書およびそのプライベートキー (どちらも PEM 形式) を使用できる必要があります。
- 証明書には、***.apps.<clustername>.<domain>** の **subjectAltName** 拡張がなければなりません。

手順

1. 新しい証明書に署名するために使用される認証局が含まれる ConfigMap を作成します。

```
$ oc create configmap custom-ca \
  --from-file=ca-bundle.crt=</path/to/example-ca.crt> \ 1
  -n openshift-config
```

- 1 **</path/to/example-ca.crt>** は、ローカルファイルシステム上の認証局ファイルへのパスです。

2. 新たに作成された ConfigMap でクラスター全体のプロキシ設定を更新します。

```
$ oc patch proxy/cluster \
  --type=merge \
  --patch='{"spec":{"trustedCA":{"name":"custom-ca"}}}'
```

3. ワイルドカード証明書およびキーが含まれるシークレットを作成します。

```
$ oc create secret tls <certificate> \ 1
  --cert=</path/to/cert.crt> \ 2
```

```
--key=</path/to/cert.key> \ 3
-n openshift-ingress
```

- 1 <certificate> は、証明書およびプライベートキーが含まれるシークレットの名前です。
- 2 </path/to/cert.crt> は、ローカルファイルシステム上の証明書へのパスです。
- 3 </path/to/cert.key> は、この証明書に関連付けられるプライベートキーへのパスです。

4. Ingress コントローラー設定を、新規に作成されたシークレットで更新します。

```
$ oc patch ingresscontroller.operator default \
  --type=merge -p \
  '{"spec":{"defaultCertificate": {"name": "<certificate>"}}}' 1
-n openshift-ingress-operator
```

- 1 <certificate> を、直前の手順のシークレットで使用された名前に置き換えます。

5.2. API サーバー証明書の追加

デフォルトの API サーバー証明書は、内部 OpenShift Container Platform クラスター CA によって発行されます。クラスター外のクライアントは、デフォルトで API サーバーの証明書を検証できません。この証明書は、クライアントが信頼する CA によって発行される証明書に置き換えることができます。

5.2.1. API サーバーの名前付き証明書の追加

デフォルトの API サーバー証明書は、内部 OpenShift Container Platform クラスター CA によって発行されます。リバースプロキシやロードバランサーが使用される場合などに、クライアントの要求 URL に基づいて送信できるように証明書を API サーバーに追加できます。

前提条件

- クライアントの URL についての PEM 形式の証明書およびキーが必要です。
- 証明書は、API サーバーに到達できるように、クライアントによって使用される URL について発行される必要があります。
- 証明書には、URL の **subjectAltName** 拡張がなければなりません。
- サーバー証明書の認証に証明書チェーンが必要な場合は、証明書チェーンをサーバー証明書に追加する必要があります。証明書ファイルは Base64 PEM でエンコードされており、通常は **.crt** または **.pem** 拡張があります。例:

```
$ cat server_cert.pem int2ca_cert.pem int1ca_cert.pem rootca_cert.pem>combined_cert.pem
```

証明書を組み合わせる際には、証明書の順序が重要になります。以下の証明書はその前にある証明書を直接認証する必要があります。たとえば、以下のようになります。

1. OpenShift Container Platform マスターホストサーバー証明書。
2. サーバー証明書を認定する中間 CA 証明書。
3. 中間 CA 証明書を認定するルート CA 証明書。



警告

内部ロードバランサーに名前付きの証明書を指定しないようにしてください (ホスト名 `api-int.<cluster_name>.<base_domain>`)。これを指定すると、クラスタの状態は動作の低下した状態になります。

手順

1. `openshift-config` namespace に証明書およびキーが含まれるシークレットを作成します。

```
$ oc create secret tls <certificate> \ 1
  --cert=</path/to/cert.crt> \ 2
  --key=</path/to/cert.key> \ 3
  -n openshift-config
```

- 1 `<certificate>` は、証明書が含まれるシークレットの名前です。
- 2 `</path/to/cert.crt>` は、ローカルファイルシステム上の証明書へのパスです。
- 3 `</path/to/cert.key>` は、この証明書に関連付けられるプライベートキーへのパスです。

2. API サーバーを作成されたシークレットを参照するように更新します。

```
$ oc patch apiserver cluster \
  --type=merge -p \
  '{"spec":{"servingCerts":{"namedCertificates":
  [{"names":["<hostname>"], 1
  "servingCertificate":{"name":"<certificate>"}}]}}' 2
```

- 1 `<hostname>` を、API サーバーが証明書を提供するホスト名に置き換えます。
- 2 `<certificate>` を、直前の手順のシークレットで使用された名前に置き換えます。

3. `apiserver/cluster` オブジェクトを検査し、シークレットが参照されていることを確認します。

```
$ oc get apiserver cluster -o yaml
...
spec:
  servingCerts:
    namedCertificates:
      - names:
        - <hostname>
      servingCertificate:
        name: <certificate>
...
```


5.3. サービス提供証明書のシークレットによるサービストラフィックのセキュリティ保護

5.3.1. サービス提供証明書について

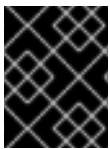
サービス提供証明書は、暗号化を必要とする複雑なミドルウェアアプリケーションをサポートすることが意図されています。これらの証明書は、TLS Web サーバー証明書として発行されます。

service-ca コントローラーは、サービス証明書を生成するために **x509.SHA256WithRSA** 署名アルゴリズムを使用します。

生成される証明書およびキーは PEM 形式のもので、作成されたシークレット内の **tls.crt** および **tls.key** にそれぞれ保存されます。証明書およびキーは、有効期間に近づくとも自動的に置き換えられます。サービス証明書を署名するサービス CA 証明書は、OpenShift Container Platform のインストール後1年間のみ有効です。

5.3.2. サービス証明書の追加

サービスとの通信のセキュリティを保護するには、サービスと同じ namespace のシークレットに署名済みの提供証明書とキーのペアを生成します。



重要

生成される証明書は、内部サービス DNS 名 **<service.name>**.
<service.namespace>.svc にのみ有効であり、内部通信用にのみ有効です。

前提条件

- サービスが定義されていること。

手順

1. サービスに **service.beta.openshift.io/serving-cert-secret-name** のアノテーションを付けます。

```
$ oc annotate service <service-name> \
  service.beta.openshift.io/serving-cert-secret-name=<secret-name>
```

- ① **<service-name>** を、セキュリティ保護するサービスの名前に置き換えます。
- ② **<secret-name>** は、証明書とキーのペアを含む生成されたシークレットの名前です。便宜上、これを **<service-name>** と同じにすることが推奨されます。

たとえば、以下のコマンドを使用してサービス **foo** にアノテーションを付けます。

```
$ oc annotate service foo service.beta.openshift.io/serving-cert-secret-name=foo
```

2. アノテーションが存在することを確認するためにサービスを検査します。

```
$ oc describe service <service-name>
...
Annotations:      service.beta.openshift.io/serving-cert-secret-name: <service-name>
```

```
service.beta.openshift.io/serving-cert-signed-by: openshift-service-serving-
signer@1556850837
...
```

3. クラスターがサービスのシークレットを生成した後に、PodSpec がこれをマウントでき、Pod はシークレットが利用可能になった後にこれを実行できます。

5.3.3. サービス証明書の ConfigMap への追加

Pod は、**service.beta.openshift.io/inject-cabundle=true** のアノテーションの付いた ConfigMap をマウントしてサービス CA 証明書にアクセスできます。アノテーションが付けられると、クラスターはサービス CA 証明書を ConfigMap の **service-ca.crt** キーに自動的に挿入します。この CA 証明書にアクセスできると、TLS クライアントはサービス提供証明書を使用してサービスへの接続を検証できます。



重要

このアノテーションが ConfigMap に追加されると、その中に含まれるすべての既存データが削除されます。**service-ca.crt** を組み込む ConfigMap としては、Pod の設定の保存先と同じ ConfigMap ではなく、別の ConfigMap を使用することが推奨されます。

手順

1. ConfigMap に **service.beta.openshift.io/inject-cabundle=true** のアノテーションを付けます。

```
$ oc annotate configmap <configmap-name> \1
service.beta.openshift.io/inject-cabundle=true
```

- 1 **<configmap-name>** を、アノテーションを付ける ConfigMap の名前に置き換えます。



注記

volumeMount の **service-ca.crt** キーを明示的に参照することにより、ConfigMap が CA バンドルと共に挿入されるまで、Pod を起動できなくなります。

たとえば、ConfigMap **foo** にアノテーションを付けるには、以下のコマンドを使用します。

```
$ oc annotate configmap foo service.beta.openshift.io/inject-cabundle=true
```

2. 証明書が生成されていることを確認するために ConfigMap を表示します。これは、YAML 出力の **service-ca.crt** として表示されます。

```
$ oc get configmap <configmap-name> -o yaml
apiVersion: v1
data:
  service-ca.crt: |
    -----BEGIN CERTIFICATE-----
...

```

5.3.4. 生成されたサービス証明書の手動によるローテーション

関連付けられたシークレットを削除することにより、サービス証明書をローテーションできます。シークレットを削除すると、新規のシークレットが自動的に作成され、新規証明書が作成されます。

前提条件

- 証明書とキーのペアを含むシークレットがサービス用に生成されていること。

手順

1. 証明書を含むシークレットを確認するためにサービスを検査します。これは、以下に示すように **servicing-cert-secret-name** アノテーションにあります。

```
$ oc describe service <service-name>
...
service.beta.openshift.io/serving-cert-secret-name: <secret>
...
```

2. サービスの生成されたシークレットを削除します。このプロセスで、シークレットが自動的に再作成されます。

```
$ oc delete secret <secret> ①
```

- ① **<secret>** を、直前の手順のシークレットの名前に置き換えます。

3. 新規シークレットを取得し、**AGE** を調べて、証明書が再作成されていることを確認します。

```
$ oc get secret <service-name>

NAME          TYPE          DATA  AGE
<service.name>  kubernetes.io/tls  2      1s
```

5.3.5. サービス CA 証明書の手動によるローテーション

サービス CA は、OpenShift Container Platform のインストール後 1 年間有効です。以下の手順に従って、有効期限前にサービス CA を手動で更新します。

前提条件

- クラスタ管理者としてログインしている必要があります。

手順

1. 以下のコマンドを使用して、現在のサービス CA 証明書の有効期限を表示します。

```
$ oc get secrets/signing-key -n openshift-service-ca \
  -o template='{{index .data "tls.crt"}}' \
  | base64 -d \
  | openssl x509 -noout -enddate
```

2. サービス CA を手動でローテーションします。このプロセスは、新規サービス証明書に署名するために使用される新規サービス CA を生成します。

```
$ oc delete secret/signing-key -n openshift-service-ca
```

3. 新規証明書をすべてのサービスに適用するには、クラスター内のすべての Pod を再起動します。このコマンドにより、すべてのサービスが更新された証明書を使用ようになります。

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



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 を調べ、これらを削除するため、サービスを中断させます。これらの Pod は削除後に自動的に再起動します。

第6章 RBAC の使用によるパーミッションの定義および適用

6.1. 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、認証とは異なる別の手順であることに注意してください。認可の手順は、アクションを実行するユーザーのアイデンティティ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。

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 を 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	ロールを使ったユーザーおよび/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。

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。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。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せん。例:

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。

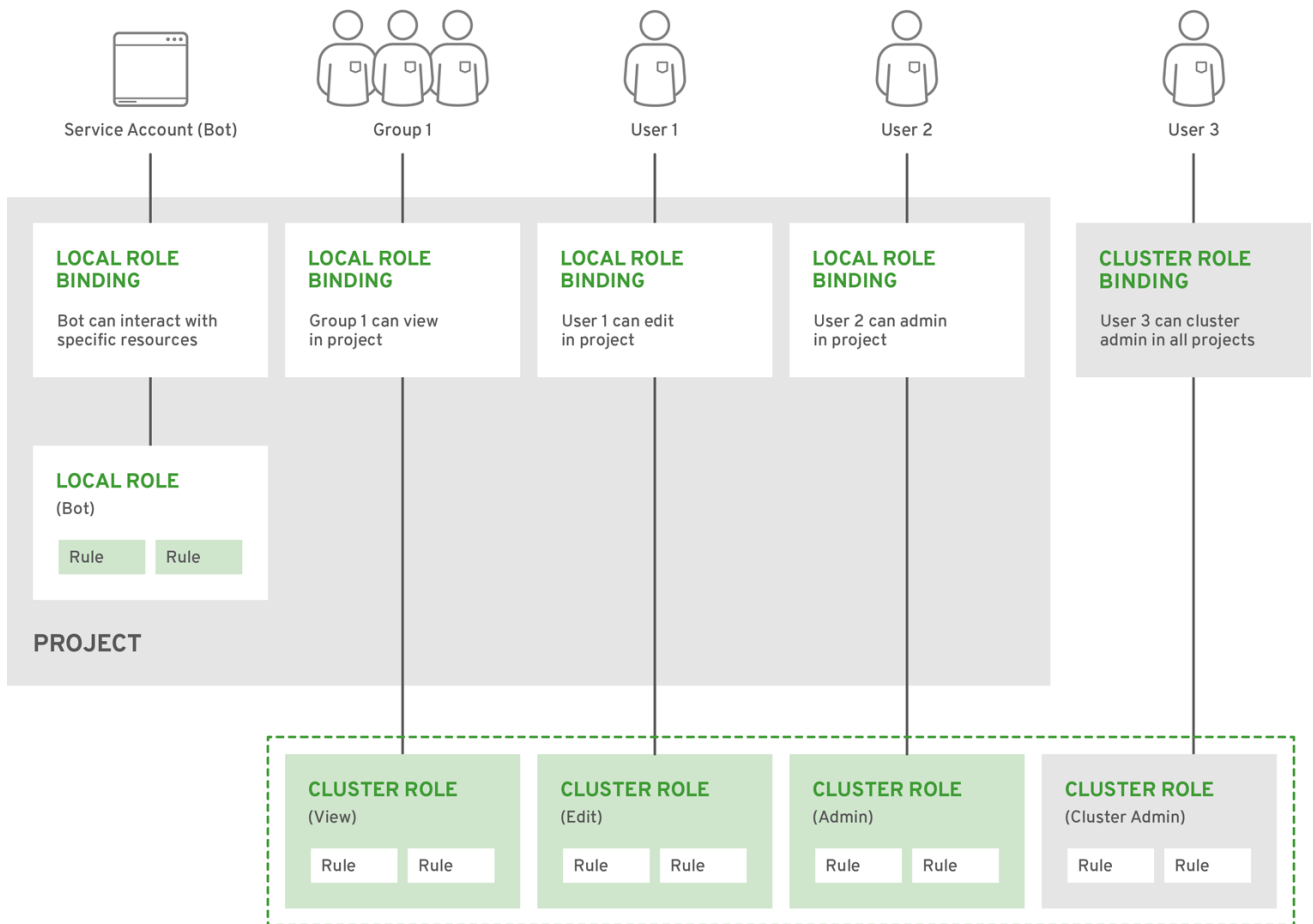
6.1.1. デフォルトのクラスターロール

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。必要な場合には、デフォルトのクラスターロールを手動で変更できますが、毎回のマスターノードの再起動時に追加の手順を実行する必要があります。

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。
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一覧表示しない 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一覧表示します。

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。



OPENSHIFT_415489_0218

6.1.2. 認可の評価

OpenShift Container Platform は以下の手順を使って認可を評価します。

アイデンティティ

ユーザーが属するユーザー名とグループの一覧。

アクション

実行する動作。ほとんどの場合、これは以下で構成されます。

- **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は、他のコミュニティ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一覧、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 は以下の手順を使って認可を評価します。

1. アイデンティティ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。

6.1.2.1. クラスターロールの集計

デフォルトのクラスターの管理、編集および cluster-reader ロールは、[クラスターロールの集計](#)をサポートします。ここでは、各ロールのクラスタールールは新規ルートの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。

6.2. 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、namespace 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が他のコミュニティ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的 ID であり、CLI ツールまたは API を使用する場合に最も明確に表示されます。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。

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 account (サービスアカウント)	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対話できます。

6.3. 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) あるとみなされ、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ます。

6.4. 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 をインストールします。
- 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。
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
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
...

```

6.5. 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 をインストールします。
- 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
```

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

6.6. 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、または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
```

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
```

```
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。

6.7. 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>
```

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カンマ区切りの一覧です。
- **<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
```

6.8. 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>
```

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カンマ区切りの一覧です。
- **<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
```

6.9. 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6.1 ローカルのロールバインディング操作

コマンド	説明
<code>\$ oc adm policy who-can <verb> <resource></code>	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。
<code>\$ oc adm policy add-role-to-user <role> <username></code>	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。
<code>\$ oc adm policy remove-role-from-user <role> <username></code>	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。
<code>\$ oc adm policy remove-user <username></code>	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。
<code>\$ oc adm policy add-role-to-group <role> <groupname></code>	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。
<code>\$ oc adm policy remove-role-from-group <role> <groupname></code>	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。
<code>\$ oc adm policy remove-group <groupname></code>	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。

6.10. 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6.2 クラスターのロールバインディング操作

コマンド	説明
<code>\$ oc adm policy add-cluster-role-to-user <role> <username></code>	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。
<code>\$ oc adm policy remove-cluster-role-from-user <role> <username></code>	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。
<code>\$ oc adm policy add-cluster-role-to-group <role> <groupname></code>	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。
<code>\$ oc adm policy remove-cluster-role-from-group <role> <groupname></code>	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。

6.11. 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7章 KUBEADMIN ユーザーの削除

7.1. 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。例:

```
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>
```

7.2. KUBEADMIN ユーザーの削除

アイデンティティ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8章 ユーザーエージェントの設定

8.1. ユーザーエージェントについて

OpenShift Container Platform は、アプリケーション開発者の CLI が OpenShift Container Platform API にアクセスすることを防ぐために使用できるユーザーエージェントを実装します。クライアントが特定のライブラリーやバイナリーファイルを使用する場合、それらのクライアントは OpenShift Container Platform API にアクセスできません。

OpenShift Container Platform CLI のユーザーエージェントを、OpenShift Container Platform 内の値のセットで構成します。

```
<command>/<version> (<platform>/<architecture>) <client>/<git_commit>
```

たとえば、以下の場合を考慮しましょう。

- <command> = **oc**
- <version> = クライアントのバージョン。例: **v4.2.0/api** で Kubernetes API に対して行われる要求が Kubernetes のバージョンを受信し、**/oapi** で OpenShift Container Platform API に対して行われる要求が OpenShift Container Platform のバージョン (**oc version** によって指定される)を受信します。
- <platform> = **linux**
- <architecture> = **amd64**
- <client> = **openshift** または **kubernetes**。要求が **/api** で Kubernetes API に対して行われるか、**/oapi** で OpenShift Container Platform API に対して行われるかによって決まります。
- <git_commit> = クライアントバージョンの Git コミット (例: **f034127**)

上記の場合、ユーザーエージェントは以下のようになります。

```
oc/v3.3.0 (linux/amd64) openshift/f034127
```

8.2. ユーザーエージェントの設定

管理者は、マスター設定の **userAgentMatching** 設定を使用してクライアントが API にアクセスできないようにすることができます。

手順

- マスター設定ファイルを、ユーザーエージェント設定を組み込むように変更します。たとえば、以下のユーザーエージェントは Kubernetes 1.2 クライアントバイナリー、OKD 1.1.3 バイナリー、および POST および PUT **httpVerbs** を拒否します。

```
policyConfig:
  userAgentMatchingConfig:
    defaultRejectionMessage: "Your client is too old. Go to https://example.org to update it."
    deniedClients:
      - regex: 'w+/v(?:?:1\.1\.1)|(?:?:1\.0\.1)) \(.+.\) openshift/w{7}'
      - regex: 'w+/v(?:?:1\.1\.3) \(.+.\) openshift/w{7}'
    httpVerbs:
```

```
- POST
- PUT
- regex: '\w+/v1\2\0 \(.+/.+)\ kubernetes/\w{7}'
httpVerbs:
- POST
- PUT
requiredClients: null
```

以下の例は、予想されるクライアントに一致しないクライアントを拒否します。

```
policyConfig:
  userAgentMatchingConfig:
    defaultRejectionMessage: "Your client is too old. Go to https://example.org to update it."
    deniedClients: []
    requiredClients:
      - regex: '\w+/v1\1\3 \(.+/.+)\ openshift/\w{7}'
      - regex: '\w+/v1\2\0 \(.+/.+)\ kubernetes/\w{7}'
    httpVerbs:
      - POST
      - PUT
```

第9章 サービスアカウントの概要および作成

9.1. 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、または削除するために API 呼び出しを実行する。
- コンテナ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以下の 2 つのグループのメンバーでもあります。

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。

9.2. 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
```

```

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> ❶

serviceaccount "robot" created

```

- ❶ 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。

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

Image pull secrets: robot-dockercfg-qzbhb

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

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

```

9.3. 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

```

- 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、**<serviceaccount_name>** を指定します。

```

$ oc policy add-role-to-user <role_name> -z <serviceaccount_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。

- 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 **top-secret** 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 top-secret
```

- **managers** プロジェクトのすべてのサービスアカウントが **top-secret** 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 top-secret
```

第10章 アプリケーションでのサービスアカウントの使用

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、または削除するために API 呼び出しを実行する。
- コンテナ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以下の2つのグループのメンバーでもあります。

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. 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。

10.2.1. 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せん。

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。

10.2.2. 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。

10.3. 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

- 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
```

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

- 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> ❶
```

```
serviceaccount "robot" created
```

- ❶ 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。

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

Image pull secrets: robot-dockercfg-qzbhb

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

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

10.4. サービスアカウントの認証情報の外部での使用

サービスアカウントのトークンは、API に対して認証する必要がある外部アプリケーションに配布することができます。

イメージをプルするには、要求される **imagestreams/layers** に対する **get** 権限が、この認証済みのユーザーに割り当てられている必要があります。また、イメージをプッシュするには、認証済みのユーザーに、要求される **imagestreams/layers** に対する **update** 権限が割り当てられている必要があります。

デフォルトで、プロジェクトのすべてのサービスアカウントは同じプロジェクトの任意のイメージをプルする権限を持ち、**builder** サービスアカウントには同じプロジェクトの任意のイメージをプッシュする権限を持ちます。

手順

1. サービスアカウントのトークンを表示します。

```
$ oc describe secret <secret-name>
```

以下は例になります。

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

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

Type: kubernetes.io/service-account-token
```

```
Data
```

```
token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9...
```

2. 取得したトークンを使用してログインします。

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

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

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

```
$ oc new-project <projectname>
```

3. サービスアカウントとしてログインしたことを確認します。

```
$ oc whoami
```

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

第11章 サービスアカウントの OAUTH クライアントとしての使用

11.1. 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>:<serviceaccount_namespace>**
- **role:<any_role>:<serviceaccount_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:<serviceaccount_namespace>:<serviceaccount_name>** になります。
- **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。例:


```
$ oc sa get-token <serviceaccount_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。

11.1.1. 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、以下のようにプレフィックス **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せん。例:

```
"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**。プレフィックスを使用した動的なリダイレクト URI を使用できます。

例:

```
"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、プレフィックスも組み合わせて、参照オブジェクトで提供されるデータを上書きできます。例:

```
"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> が有効とみなされますが、<https://example.com> は有効とみなされません。上書き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になります。

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



注記

ホスト名の上書きを指定すると、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って実行できます。

<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12章 スコープトークン

12.1. 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」の承認者に渡せるよう、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。

12.1.1. 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一覧表示するための読み取り専用アクセスを許可します。

12.1.2. 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つことができます。

- **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ることが多いですが、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13章 SCC (SECURITY CONTEXT CONSTRAINTS) の管理

13.1. SCC (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 **SCC (Security Context Constraints)** を使用して Pod のパーミッションを制御できます。これらのパーミッションには、コンテナのコレクションである **pod** が実行できるアクションおよびそれがアクセスできるリソース情報が含まれます。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 の実行についての条件の一覧を定義することができます。

SCC により、管理者は以下を制御できます。

- Pod が特権付きコンテナを実行できるかどうか。
- コンテナが要求できる機能。
- ホストディレクトリーのボリュームとしての使用。
- コンテナの SELinux コンテキスト
- コンテナのユーザー ID。
- ホストの namespace およびネットワークの使用
- Pod の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。
- コンテナが読み取り専用のルートファイルシステムの使用を要求するかどうか。
- 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。

Docker には、Pod の各コンテナについて許可される [デフォルトの機能一覧](#) があります。コンテナはこれらの機能をデフォルト一覧から使用しますが、Pod マニフェストの作成者は追加機能を要求したり、デフォルトからデフォルト動作の一部を削除してこの一覧を変更できます。 **allowedCapabilities**、 **defaultAddCapabilities**、 および **requiredDropCapabilities** パラメーターは Pod からのこのような要求を制御し、要求できる機能を決定し、各コンテナに追加するものや禁止する必要のあるものを決定するために使用されます。

クラスターには、8 つのデフォルト SCC が含まれます。

- **anyuid**
- **hostaccess**
- **hostmount-anyuid**
- **hostnetwork**



警告

追加のワークロードをマスターホストで実行する場合、**hostnetwork** へのアクセスを提供する際には注意して行ってください。マスターホストで **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、適切に信頼される必要があります。

- **node-exporter**
- **nonroot**
- **privileged**
- **restricted**



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、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。代わりに新規の SCC を作成してください。

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
- Pods 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

restricted SCC は以下を実行します。

- Pod が特権付き Pod として実行されないようにします。
- Pod がホストディレクトリーの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 の範囲でユーザーとして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 MCS ラベルで実行されることを要求します。
- Pod が FSGroup を使用することを許可します。
- Pod が補助グループを使用することを許可します。



注記

それぞれの SCC についての詳細は、SCC で利用可能な kubernetes.io/description アノテーションを参照してください。

SCC は Pod がアクセスできるセキュリティ機能を制限する各種の設定およびストラテジーで構成されています。これらの設定は以下のカテゴリーに分類されます。

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 は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はセットに対してチェックされ、それら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。 <ul style="list-style-type: none"> • 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

13.1.1. SCC ストラテジー

RunAsUser

1. **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。
2. **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。
3. **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。
4. **RunAsAny:** デフォルトは指定されません。**runAsUser** の指定を許可します。

SELinuxContext

1. **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。
2. **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

1. **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。
2. **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

1. **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。
2. **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。

13.1.2. 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。

- [azureFile](#)
- [azureDisk](#)
- [flocker](#)
- [flexVolume](#)
- [hostPath](#)
- [emptyDir](#)
- [gcePersistentDisk](#)
- [awsElasticBlockStore](#)
- [gitRepo](#)
- [secret](#)
- [nfs](#)
- [iscsi](#)
- [glusterfs](#)
- [persistentVolumeClaim](#)
- [rbd](#)
- [cinder](#)
- [cephFS](#)
- [downwardAPI](#)
- [fc](#)

- **configMap**
- **vsphereVolume**
- **quobyte**
- **photonPersistentDisk**
- **projected**
- **portworxVolume**
- **scaleIO**
- **storageos**
- * (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します)

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一覧は、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、網羅的な一覧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上書きします。たとえば、**allowHostDirVolumePlugin** が **false** に設定されるが、**volumes** フィールドで許可される場合、**hostPath** 値は **volumes** から削除されません。

13.1.3. 受付 (Admission)

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、許可される SCC のセットには、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。

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の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、Pod が受け入れられます。要求が SCC に一致しない場合、Pod は拒否されます。

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、検証する必要がある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る値を使用するストラテジーに関連するものです。

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 は 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。

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 は 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。

13.1.4. SCC の優先度設定

SCC には、受付コントローラーによる要求の検証を試行する際の順序に影響を与える優先度フィールドがあります。優先度の高い SCC は並び替える際にセットの先頭に移動します。利用可能な SCC の完全なセットが判別されると、それらは以下に戻づいて順序付けられます。

1. 優先度が高い順。nil は優先度 0 とみなされます。
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. 優先度と制限のどちらも等しい場合、SCC は名前順に並べ替えられます。

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せん。これにより、クラスター管理者は Pod の **SecurityContext** で **RunAsUser** を指定しなくても Pod を任意のユーザーとして実行できます。管理者は、希望する場合は依然として **RunAsUser** を指定できます。

13.2. 事前に割り当てられる SCC (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 SCC (Security Context Constraints) 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個に評価されます。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、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、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。単一の値の **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、SCC は作成されません。

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、自ら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** を読み取る場合、**FSGroup** ストラテジーは **1** の最小値および最大値で自らを設定します。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、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>` 形式のカンマ区切りのブロックの一覧を受け入れます。`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せん。

13.3. SCC (SECURITY CONTEXT CONSTRAINTS) の例

以下の例は、SCC (Security Context Constraint) 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: 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
runAsUser: ⑥
  type: RunAsAny
seLinuxContext: ⑦
  type: RunAsAny
seccompProfiles:
- '*'
supplementalGroups: ⑧
  type: RunAsAny
users: ⑨
- system:serviceaccount:default:registry
- system:serviceaccount:default:routier
- system:serviceaccount:openshift-infra:build-controller
volumes:
- '*'
```

- 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、一覧が空の場合は、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一覧です。
- 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 からドロップされる機能の一覧です。
- 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。

SCC の **users** 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は特権付き SCC へのアクセスが付与されます。認証されるすべてのユーザーには制限付き 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 によって異なります。制限付き 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。この制限付き 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 ❶
containers:
  - name: sec-ctx-demo
    image: gcr.io/google-samples/node-hello:1.0
```

- ❶ 特定のユーザー ID を要求するコンテナまたは 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。

13.4. SCC (SECURITY CONTEXT CONSTRAINTS) の作成

CLI を使用して SCC (Security Context Constraint) を作成することができます。

前提条件

- **oc** コマンドラインをインストールする必要があります。
- アカウントには SCC を作成できるように **cluster-admin** 権限がなければなりません。

手順

1. JSON または YAML ファイルで SCC を定義します。

SCC (Security Context Constraints) オブジェクトの定義

```
kind: SecurityContextConstraints
apiVersion: 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ドロップ機能を SCC に追加することができます。指定された機能はコンテナからドロップされることとなります。たとえば、SCC を **KILL**、**MKNOD**、および **SYS_CHROOT** の必要なドロップ機能を使って作成するには、以下を SCC オブジェクトに追加します。

```
requiredDropCapabilities:
  - KILL
```

```
- MKNOD
- SYS_CHROOT
```

使用できる値の一覧は、[Docker ドキュメント](#)で確認できます。

ヒント

機能は Docker に渡されるため、特殊な **ALL** 値を使用してすべての機能をドロップすることができます。

- 次に、作成するファイルを渡して **oc create** を実行します。

```
$ oc create -f scc_admin.yaml
securitycontextconstraints "scc-admin" created
```

- SCC が作成されていることを確認します。

```
$ oc get scc scc-admin
NAME      PRIV  CAPS  SELINUX  RUNASUSER  FSGROUP  SUPGROUP
PRIORITY  READONLYROOTFS  VOLUMES
scc-admin true  []    RunAsAny RunAsAny  RunAsAny RunAsAny <none>  false
[jawsElasticBlockStore azureDisk azureFile cephFS cinder configMap downwardAPI
emptyDir fc flexVolume flocker gcePersistentDisk gitRepo glusterfs iscsi nfs
persistentVolumeClaim photonPersistentDisk quobyte rbd secret vsphere]
```

13.5. SCC (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 へのアクセスの範囲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、クラスター全体の範囲が保持されます。



注記

SCC を次のデフォルト namespace のいずれかに作成します。 **default**、**kube-system**、**kube-public**、**openshift-node**、**openshift-infra**、**openshift**。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 SecurityContextConstraint リソースが含まれる 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一覧。

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、RoleBinding または ClusterRoleBinding 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** 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。

13.6. SCC (SECURITY CONTEXT CONSTRAINTS) 参照コマンド

CLI を使用して、インスタンス内の SCC 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。



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、アップグレード時に問題が生じる可能性があります。代わりに新規の SCC を作成してください。

13.6.1. SCC の一覧表示

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
```

NAME	PRIV	CAPS	SELINUX	RUNASUSER	FSGROUP	SUPGROUP
anyuid	false	[]	MustRunAs	RunAsAny	RunAsAny	RunAsAny 10 false
[configMap downwardAPI emptyDir persistentVolumeClaim projected secret]						
hostaccess	false	[]	MustRunAs	MustRunAsRange	MustRunAs	RunAsAny <none>
false						[configMap downwardAPI emptyDir hostPath persistentVolumeClaim projected secret]
hostmount-anyuid	false	[]	MustRunAs	RunAsAny	RunAsAny	RunAsAny <none>
false						[configMap downwardAPI emptyDir hostPath nfs persistentVolumeClaim projected secret]
hostnetwork	false	[]	MustRunAs	MustRunAsRange	MustRunAs	MustRunAs <none>
false						[configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
node-exporter	false	[]	RunAsAny	RunAsAny	RunAsAny	RunAsAny <none> false
[*]						
nonroot	false	[]	MustRunAs	MustRunAsNonRoot	RunAsAny	RunAsAny <none>
false						[configMap downwardAPI emptyDir persistentVolumeClaim projected secret]
privileged	true	[*]	RunAsAny	RunAsAny	RunAsAny	RunAsAny <none> false
[*]						
restricted	false	[]	MustRunAs	MustRunAsRange	MustRunAs	RunAsAny <none>
false						[configMap downwardAPI emptyDir persistentVolumeClaim projected secret]

13.6.2. SCC の検査

特定の SCC についての情報（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
Name: restricted
Priority: <none>
Access:
  Users: <none> 1
  Groups: system:authenticated 2
Settings:
  Allow Privileged: false
  Default Add Capabilities: <none>
  Required Drop Capabilities: KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities: <none>
  Allowed Seccomp Profiles: <none>
  Allowed Volume Types:
configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,secret
  Allow Host Network: false
  Allow Host Ports: false
  Allow Host PID: false
  Allow Host IPC: false
  Read Only Root Filesystem: false
  Run As User Strategy: MustRunAsRange
  UID: <none>
  UID Range Min: <none>
  UID Range Max: <none>
SELinux Context Strategy: MustRunAs
  User: <none>
  Role: <none>
  Type: <none>
```

```
Level: <none>  
FSGroup Strategy: MustRunAs  
Ranges: <none>  
Supplemental Groups Strategy: RunAsAny  
Ranges: <none>
```

- 1 SCC が適用されるユーザーとサービスアカウントを一覧表示します。
- 2 SCC が適用されるグループを一覧表示します。



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。

13.6.3. SCC 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、それはクラスタの再起動時に再生成されます。

13.6.4. SCC 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。

第14章 SYSTEM:ADMIN ユーザーの権限の借用

14.1. 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。

14.2. 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>
```

第15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。

15.1. 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一覧です。

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、**ldaps://** URL を指定しない場合は

- 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、同期に必要なエントリーの LDAP クエリー定義で構成されています。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=inetOrgPerson) 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15.1 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15.2 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
```

15.1.1. 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 ❶
insecure: false ❷
rfc2307:
  groupsQuery:
    baseDN: "ou=groups,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  groupUIDAttribute: dn ❸
  groupNameAttributes: [ cn ] ❹
  groupMembershipAttributes: [ member ] ❺
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  userUIDAttribute: dn ❻
  userNameAttributes: [ mail ] ❼
tolerateMemberNotFoundErrors: false
tolerateMemberOutOfScopeErrors: false

```

- ❶ 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。
- ❷ **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、 **ldaps://** URL を指定しない場合はサーバーへの TLS 接続は行われません。指定している場合は、URL は TLS を使用して接続を試行します。
- ❸ 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。
- ❹ Group の名前として使用する属性です。
- ❺ メンバーシップ情報を保存するグループの属性です。
- ❻ 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。
- ❼ 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。

15.1.2. 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=inetOrgPerson)
    pageSize: 0
  userNameAttributes: [ mail ] ❶
  groupMembershipAttributes: [ memberOf ] ❷

```

- ❶ 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。
- ❷ メンバーシップ情報を保存するユーザーの属性です。

15.1.3. 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 ❶
  groupNameAttributes: [ cn ] ❷
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    filter: (objectclass=inetOrgPerson)
    pageSize: 0
  userNameAttributes: [ mail ] ❸
  groupMembershipAttributes: [ memberOf ] ❹

```

- 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 Group の名前として使用する属性です。
- 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。

15.2. 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。

15.2.1. 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

1. 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。

15.2.2. 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

1. 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。

15.2.3. 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ます。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せん。

前提条件

- 同期設定ファイルを作成します。

手順

1. 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。

15.3. 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。

例:

```
$ 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
```

15.4. LDAP グループの同期の例

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



注記

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

15.4.1. RFC 2307 スキーマの使用によるグループの同期

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

- グループとユーザーが LDAP サーバーに追加される方法。
- 同期後に生成される 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 ❶
objectClass: groupOfNames
cn: admins
owner: cn=admin,dc=example,dc=com
description: System Administrators
member: cn=Jane,ou=users,dc=example,dc=com ❷
member: cn=Jim,ou=users,dc=example,dc=com

```

- ❶ 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。
- ❷ 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一覧表示されます。

前提条件

- 設定ファイルを作成します。

手順

1. `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 ❶
    openshift.io/ldap.uid: cn=admins,ou=groups,dc=example,dc=com ❷
    openshift.io/ldap.url: LDAP_SERVER_IP:389 ❸

```

```

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。

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

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

ユーザー定義の名前マッピングに関する RFC 2307 スキーマを使用する LDAP 同期設定:
rfc2307_config_user_defined.yaml

```

kind: LDAPSyncConfig
apiVersion: v1
groupUIDNameMapping:
  "cn=admins,ou=groups,dc=example,dc=com": Administrators ①
rfc2307:
  groupsQuery:
    baseDN: "ou=groups,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  groupUIDAttribute: dn ②
  groupNameAttributes: [ cn ] ③
  groupMembershipAttributes: [ member ]
  usersQuery:
    baseDN: "ou=users,dc=example,dc=com"
    scope: sub
    derefAliases: never
    pageSize: 0
  userUIDAttribute: dn ④
  userNameAttributes: [ mail ]
  tolerateMemberNotFoundErrors: false
  tolerateMemberOutOfScopeErrors: false

```

① ユーザー定義の名前マッピングです。

②

ユーザー定義の名前マッピングでキーに使用される固有の識別属性です。groupUIDAttribute に DN を使用している場合は **groupsQuery** フィルターを指定できません。詳細なフィルターを実行

- 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

1. **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。

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

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

```
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with dn="<user-dn">" would search outside of the base dn specified (dn="<base-dn">)".
```

これは **usersQuery** フィールドの **baseDN** の設定が間違っていることを示していることがよくありま

す。ただし、**baseDN** にグループの一部のメンバーが意図的に含まれていない場合、**tolerateMemberOutOfScopeErrors: true** を設定することでグループ同期が継続されます。範囲外のメンバーは無視されます。

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

```
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with base dn="<user-dn>" refers to a non-existent entry".
```

```
Error determining LDAP group membership for "<group>": membership lookup for user "<user>" in group "<group>" failed because of "search for entry with base dn="<user-dn>" and filter "<filter>" did not return any results".
```

これは **usersQuery** フィールドの設定が間違っていることを示していることがよくあります。ただし、グループに欠落していると認識されているメンバーエントリが含まれる場合、**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: 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 ❶
member: cn=Jim,ou=OUTOFSCOPE,dc=example,dc=com ❷

```

- ❶ LDAP サーバーに存在しないメンバーです。
- ❷ 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: LDAPSvcConfig
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 ❶
  userNameAttributes: [ mail ]
  tolerateMemberNotFoundErrors: true ❷
  tolerateMemberOutOfScopeErrors: true ❸

```

- ❶ 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。
- ❷ **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。
- ❸ **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

1. `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。

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

Active Directory スキーマでは、両方のユーザー (Jane と Jim) がファーストクラスエントリーとして LDAP サーバーに存在し、グループメンバーシップはユーザーの属性に保存されます。以下の `Idif` 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.Idif`

```
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 ❶
```

```
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
```

- ❶ 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一覧表示され、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

1. `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 ❶
    openshift.io/ldap.uid: admins ❷
    openshift.io/ldap.url: LDAP_SERVER_IP:389 ❸
  creationTimestamp:
    name: admins ❹
  users: ❺
  - jane.smith@example.com
  - jim.adams@example.com
```

- ❶ 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。
- ❷ LDAP サーバーのグループの固有識別子です。
- ❸ このグループのレコードが保存される LDAP サーバーの IP アドレスとポートです。

- 4 LDAP サーバーに一覧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。

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

拡張された Active Directory スキーマでは、両方のユーザー (Jane と Jim) とグループがファーストクラスエントリとして 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: `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=admin,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=admin,ou=groups,dc=example,dc=com

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

dn: cn=admin,ou=groups,dc=example,dc=com 2
objectClass: groupOfNames
cn: admin
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一覧表示されます。
- 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です。

前提条件

- 設定ファイルを作成します。

手順

1. `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。

```
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 アドレスとポートです。
- 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。

15.4.5.1. 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: **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一覧表示されます。
- 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=inetOrgPerson)
    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 とスコープ値は無視されます。 **groupsQuery** 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 Group の名前として使用する属性です。
- 4 OpenShift Container Platform Group レコードでユーザーの名前として使用する属性です。ほとんどのインストールで、 **mail** または **sAMAccountName** を選択することが推奨されています。
- 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

1. **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る OpenShift 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。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。

15.5. LDAP 同期設定の仕様

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



重要

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

15.5.1. v1.LDAPSyncConfig

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

名前	説明	スキーマ
kind	このオブジェクトが表す REST リソースを表す文字列の値です。サーバーはクライアントが要求を送信するエンドポイントからこれを推測できることがあります。これを更新することはできません。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

名前	説明	スキーマ
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 に設定し、 ldaps://URL スキームを使用する場合、URL は指定された ca を使用して TLS 接続を試行します。	ブール値
ca	サーバーへ要求を行う際に使用する任意の信頼された認証局バンドルです。空の場合、デフォルトのシステムルートが使用されます。	文字列
groupUIDNameMapping	LDAP グループ UID 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

名前	説明	スキーマ
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を抽出し、グループメンバーシップはメンバーを一覧表示するグループエントリ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一覧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

15.5.2. v1.StringSource

StringSource によって文字列インラインを指定できます。または環境変数またはファイルを使用して外部から指定することもできます。文字列の値のみを含む場合、単純な JSON 文字列にマーシャルします。

名前	説明	スキーマ
value	クリアテキスト値、または keyFile が指定されている場合は暗号化された値を指定します。	文字列
env	クリアテキスト値、または keyFile が指定されている場合は暗号化された値を含む環境変数を指定します。	文字列

名前	説明	スキーマ
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

15.5.3. v1.LDAPQuery

LDAPQuery は LDAP クエリーの作成に必要なオプションを保持します。

名前	説明	スキーマ
baseDN	すべての検索が開始されるディレクトリーのブランチの DN です。	文字列
scope	検索の任意の範囲です。 base (ベースオブジェクトのみ)、 one (ベースレベルのすべてのオブジェクト)、 sub (サブツリー全体) のいずれかになります。設定されていない場合は、デフォルトで sub になります。	文字列
derefAliases	エイリアスに関する検索の任意の動作です。 never (エイリアスを逆参照しない)、 search (検索中の逆参照のみ)、 base (ベースオブジェクト検索時の逆参照のみ)、 always (常に逆参照を行う) のいずれかになります。設定されていない場合、デフォルトで always になります。	文字列
timeout	応答の待機を中止するまでにサーバーへの要求を未処理のままにする時間制限 (秒単位) を保持します。これが 0 の場合、クライアント側の制限が設定されないことになります。	整数
filter	ベース DN を持つ LDAP サーバーから関連するすべてのエントリーを取得する有効な LDAP 検索フィルターです。	文字列

名前	説明	スキーマ
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

15.5.4. v1.RFC2307Config

RFC2307Config は、RFC2307 スキーマを使用してどのように LDAP グループ同期が LDAP サーバーに相互作用するかを定義するために必要な設定オプションを保持します。

名前	説明	スキーマ
groupsQuery	グループエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery
groupUIDAttribute	LDAP グループエントリーのどの属性が固有の識別子として解釈されるかを定義します。 (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。	文字列の配列
groupMembershipAttributes	LDAP グループエントリーのどの属性がメンバーとして解釈されるかを定義します。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。	文字列の配列
usersQuery	ユーザーエントリーを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery
userUIDAttribute	LDAP ユーザーエントリーのどの属性が固有の識別子として解釈されるかを定義します。 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

名前	説明	スキーマ
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の名前として使用する属性です。ほとんどのインストールで、 mail または sAMAccountName を選択することが推奨されています。	文字列の配列
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
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

15.5.5. v1.ActiveDirectoryConfig

ActiveDirectoryConfig は必要な設定オプションを保持し、どのように LDAP グループ同期が Active Directory スキーマを使用して LDAP サーバーと相互作用するかを定義します。

名前	説明	スキーマ
usersQuery	ユーザーエントリを返す 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の名前として使用する属性です。ほとんどのインストールで、 mail または sAMAccountName を選択することが推奨されています。	文字列の配列
groupMembershipAttributes	LDAP ユーザーのどの属性がメンバーの属するグループとして解釈されるかを定義します。	文字列の配列

15.5.6. v1.AugmentedActiveDirectoryConfig

AugmentedActiveDirectoryConfig は必要な設定オプションを保持し、どのように LDAP グループ同期が拡張された Active Directory スキーマを使用して LDAP サーバーに相互作用するかを定義します。

名前	説明	スキーマ
usersQuery	ユーザーエントリを返す 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の名前として使用する属性です。ほとんどのインストールで、 mail または sAMAccountName を選択することが推奨されています。	文字列の配列
groupMembershipAttributes	LDAP ユーザーのどの属性がメンバーの属するグループとして解釈されるかを定義します。	文字列の配列

名前	説明	スキーマ
groupsQuery	グループエントリを返す LDAP クエリーのテンプレートを保持します。	v1.LDAPQuery
groupUIDAttribute	LDAP グループエントリのどの属性が固有の識別子として解釈されるかを定義します。 (IdapGroupUID)	文字列
groupNameAttributes	LDAP グループエントリのどの属性が 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。	文字列の配列

第16章 追加ホストから API サーバーへの JAVASCRIPT ベースのアクセスの許可

16.1. 追加ホストから API サーバーへの JAVASCRIPT ベースのアクセスの許可

デフォルトの OpenShift Container Platform 設定は、OpenShift Web コンソールが要求を API サーバーに送信することのみを許可します。

別の名前を使用して JavaScript アプリケーションから API サーバーまたは OAuth サーバーにアクセスする必要がある場合、許可する追加のホスト名を設定できます。

前提条件

- **cluster-admin** ロールを持つユーザーとしてのクラスターへのアクセスがあること。

手順

1. API サーバーリソースを編集します。

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

2. **additionalCORSAAllowedOrigins** フィールドを **spec** セクションの下に追加し、1つ以上の追加のホスト名を指定します。

```
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  annotations:
    release.openshift.io/create-only: "true"
  creationTimestamp: "2019-07-11T17:35:37Z"
  generation: 1
  name: cluster
  resourceVersion: "907"
  selfLink: /apis/config.openshift.io/v1/apiservers/cluster
  uid: 4b45a8dd-a402-11e9-91ec-0219944e0696
spec:
  additionalCORSAAllowedOrigins:
    - (?i)//my\.subdomain\.domain\.com(:|z) ❶
```

- ❶ ホスト名は [Golang 正規表現](#) として指定されます。これは、API サーバーおよび OAuth サーバーに対する HTTP 要求の CORS ヘッダーに対するマッチングを行うために使用されます。



注記

この例では、以下の構文を使用します。

- **(?i)** は大文字/小文字を区別します。
- **//** はドメインの開始にピニングし、**http:** または **https:** の後のダブルスラッシュに一致します。
- **\.** はドメイン名のドットをエスケープします。
- **(:|z)** はドメイン名 (**z**) またはポートセパレーター (**:**) の終了部に一致します。

3. 変更を適用するためにファイルを保存します。