OpenShift Dedicated 4

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

Last Updated: 2020-05-15

OpenShift Dedicated 4 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

概要

本書では、OpenShift Dedicated 4 でアイデンティティープロバイダーを定義する方法を説明します。

目次

第1章 認証について

ユーザーが OpenShift Dedicated 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 Dedicated 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。

1.1. ユーザー

OpenShift Dedicated の **ユーザー** は、OpenShift Dedicated API への要求を実行できるエンティティーです。OpenShift Dedicated 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 Dedicated 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 Dedicated 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 Dedicated 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 Dedicated 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 Dedicated 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 Dedicated OAuth サーバー

OpenShift Dedicated マスターには、ビルトイン OAuth サーバーが含まれます。ユーザーは OAuth アクセストークンを取得して、API に対して認証します。

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

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

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 Dedicated 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、認証プロキシーをこのエンドポイントの前に配置するか、または OpenShift Dedicated を、サポートするアイデンティティープロバイダーに対して認証情報を検証するように設定します。**<namespace_route>/oauth/authorize** の要求は、CLI などの対話式ログインページを表示できないユーザーエージェントから送られる場合があります。そのため、OpenShift Dedicated は、対話式ログインフローのほかにも **WWW-Authenticate** チャレンジを使用した認証をサポートします。

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

注記

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

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

第2章 アイデンティティープロバイダー設定について

2.1. アイデンティティープロバイダーパラメーター

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

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

注記

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

2.2. サポートされるアイデンティティープロバイダー

以下の種類のアイデンティティープロバイダーを設定できます。

アイデンティティープロバイダー	説明
LDAP	**ldap** アイデンティティープロバイダーを、単純なバインド認証を使用して LDAPv3 サーバーに対してユーザー名とパスワードを検証するように設定します。
GitHub または GitHub Enterprise	**github** アイデンティティープロバイダーを、GitHub または GitHub Enterprise の OAuth 認証サーバーに対してユーザー名とパスワードを検証するように設定します。
Google	**google** アイデンティティープロバイダーを、Google の OpenID Connect 統合を使用して設定します。
OpenID Connect	**oidc** アイデンティティープロバイダーを、Authorization Code Flowを使用して OpenID Connect アイデンティティープロバイダーと統合するように設定します。

第3章 アイデンティティープロバイダーの設定

3.1. LDAP アイデンティティープロバイダーの設定

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

3.1.1. LDAP 認証について

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

以下の手順が実行されます。

1. 設定された **url** の属性およびフィルターとユーザーが指定したユーザー名を組み合わせて検索フィルターを生成します。
2. 生成されたフィルターを使用してディレクトリーを検索します。検索によって 1 つもエントリーが返されない場合は、アクセスを拒否します。
3. 検索で取得したエントリーの DN とユーザー指定のパスワードを使用して LDAP サーバーへのバインドを試みます。
4. バインドが失敗した場合は、アクセスを拒否します。
5. バインドが成功した場合は、アイデンティティー、電子メールアドレス、表示名、および推奨ユーザー名として設定された属性を使用してアイデンティティーを作成します。

設定される **url** は、LDAP ホストと使用する検索パラメーターを指定する RFC 2255 URL です。URL の構文は以下のようになります。

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

この URL の場合:

URL コンポーネント	説明
ldap	通常の LDAP の場合は、文字列 **ldap** を使用します。セキュアな LDAP (LDAPS) の場合は、代わりに **ldaps** を使用します。
host:port	LDAP サーバーの名前とポートです。デフォルトは、ldap の場合は **localhost:389**、LDAPS の場合は **localhost:636** です。
basedn	すべての検索が開始されるディレクトリーのブランチの DN です。これは少なくともディレクトリーツリーの最上位になければなりませんが、ディレクトリーのサブツリーを指定することもできます。

URL コンポーネント	説明
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)
```

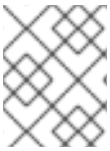
クライアントが **bob** というユーザー名を使用して接続を試みる場合、生成される検索フィルターは **(&(enabled=true)(cn=bob))** になります。

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。

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

github アイデンティティープロバイダーを、GitHub または GitHub Enterprise の OAuth 認証サーバーに対してユーザー名とパスワードを検証するように設定します。OAuth は OpenShift Dedicated 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 Dedicated 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 Dedicated クラスターにログインできないようにするために、アクセスを特定の GitHub 組織のユーザーに制限することができます。

3.2.1. GitHub アプリケーションの登録

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

手順

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

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

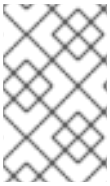
   例:

   ```
   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、アイデンティティープロバイダーの設定を完了するために必要です。

3.3. GOOGLE アイデンティティープロバイダーの設定

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



注記

Google をアイデンティティープロバイダーとして使用するには、**<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

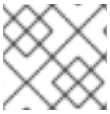
Google をアイデンティティープロバイダーとして使用することで、Google ユーザーはサーバーに対して認証されます。**hostedDomain** 設定属性を使用して、特定のホストドメインのメンバーに認証を限定することができます。

3.4. OPENID CONNECT アイデンティティープロバイダーの設定

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

重要

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



注記

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

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

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

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

また、どの要求をユーザーの推奨ユーザー名、表示名およびメールアドレスとして使用するか指定することができます。複数の要求が指定されている場合、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ー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ーです。
preferred_username	ユーザーのプロビジョニング時に優先されるユーザー名です。**janedoe** などのユーザーを参照する際に使用する省略形の名前です。通常は、ユーザー名またはメールなどの、認証システムのユーザーのログインまたはユーザー名に対応する値です。
email	メールアドレス。
name	表示名。

詳細は、OpenID claim のドキュメント を参照してください。

注記

OpenID Connect アイデンティティー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。

第4章 RBAC の使用によるパーミッションの定義および適用

4.1. 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。

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ーの判別により密接に関連しています。

認可は以下を使用して管理されます。

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す。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。

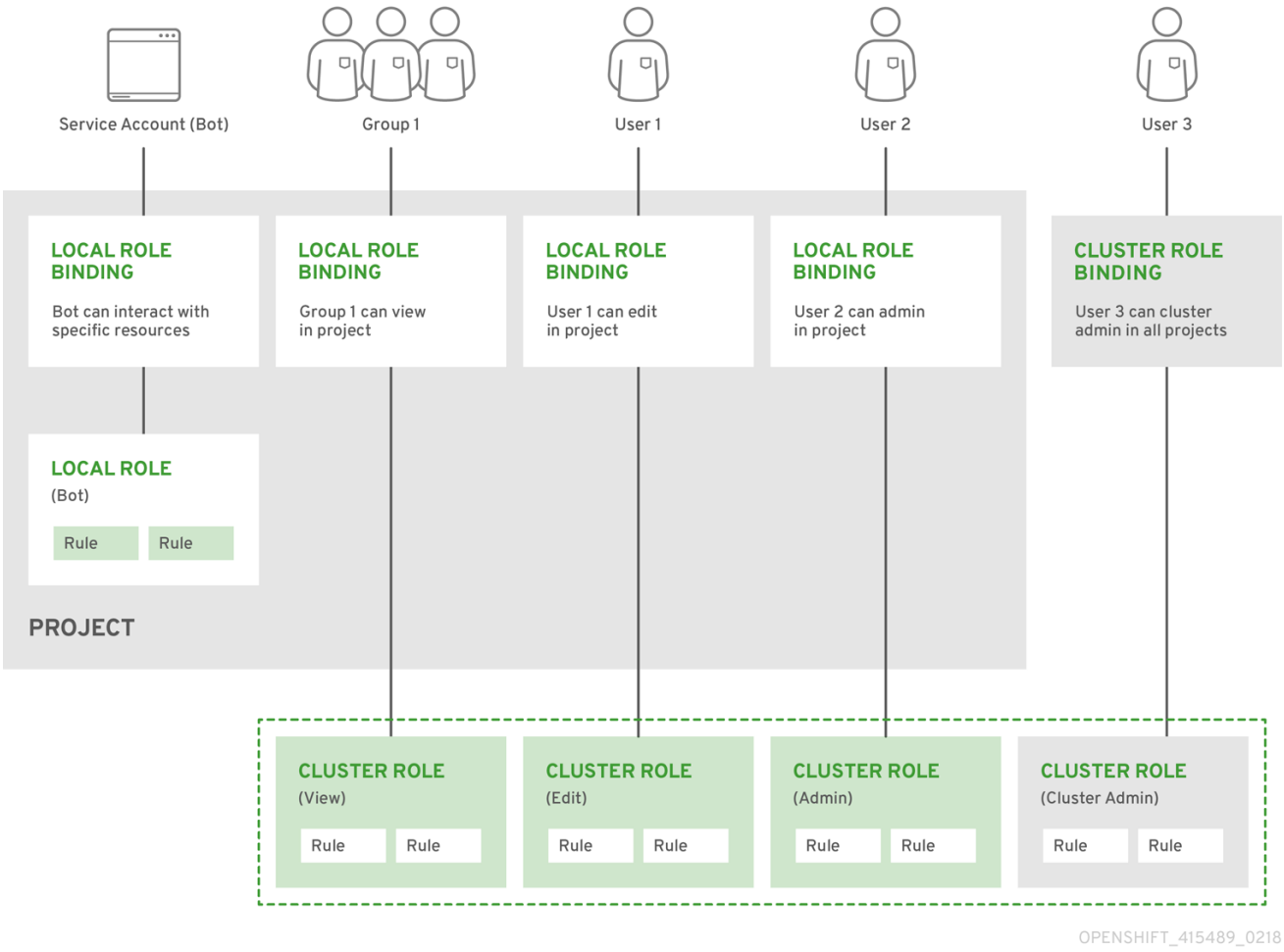
4.1.1. デフォルトのクラスターロール

OpenShift Dedicated 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。



4.1.2. 認可の評価

OpenShift Dedicated は以下を使用して認可を評価します。

アイデンティティー

ユーザーが属するユーザー名とグループの一覧。

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ーは、他のコミュニティーと分離された状態で独自のコンテンツを編成し、管理できます。
- **動詞**: **get**、**list**、 **create**、**update**、 **delete**、 **deletecollection**、または **watch** などのアクション自体。
- **リソース名**: アクセスする API エンドポイント。

バインディング

バインディングの詳細な一覧、ロールを持つユーザーまたはグループ間の関連付け。

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

1. アイデンティティーおよびプロジェクトでスコープ設定されたアクションは、ユーザーおよびそれらのグループに適用されるすべてのバインディングを検索します。

2. バインディングは、適用されるすべてのロールを見つけるために使用されます。
3. ロールは、適用されるすべてのルールを見つけるために使用されます。
4. 一致を見つけるために、アクションが各ルールに対してチェックされます。
5. 一致するルールが見つからない場合、アクションはデフォルトで拒否されます。

ヒント

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

プロジェクト管理者は CLI を使用してローカル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。

4.2. プロジェクトおよび NAMESPACE

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

Namespace は以下の一意のスコープを提供します。

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

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

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

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

- 必須の **name** はプロジェクトの一意の 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ーの任意のメンバーに委任できます。クラスター管理者は、開発者が独自のプロジェクトを作成することも許可できます。

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

4.3. デフォルトプロジェクト

OpenShift Dedicated 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。

4.4. 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。**dedicated-admins-cluster** 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を表示できます。

手順

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

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

1. 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
```

4.5. 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。
  - **dedicated-admins-cluster** ロールを持つユーザーは、ローカルロールとバインディングを表示し、管理できます。
  - 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
```

4.6. 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。

Dedicated の管理者はクラスターロールを管理できません。これらの管理者は、クラスターのロールバインディングおよびローカル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。

4.7. 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4.1 ローカルのロールバインディング操作

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

コマンド	説明
$ oc adm policy add-role-to-group <role> <groupname>	指定されたロールを現在のプロジェクトの指定グループにバインドします。
$ oc adm policy remove-role-from-group <role> <groupname>	現在のプロジェクトの指定グループから指定されたロールを削除します。
$ oc adm policy remove-group <groupname>	現在のプロジェクトの指定グループとそれらのロールのすべてを削除します。

第5章 サービスアカウントの概要および作成

5.1. 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 Dedicated 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 Dedicated 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。
- コンテナー内のアプリケーションが検出目的で API 呼び出しを実行する。
- 外部アプリケーションがモニターまたは統合目的で API 呼び出しを実行する。

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

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

すべてのサービスアカウントは以下の 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。

5.2. OPENSHIFT DEDICATED のサービスアカウント

管理者として、サービスアカウントを使用して OpenShift Dedicated の **admin** ロールを必要とするすべてのアクションを実行できます。

dedicated-admin サービスは、**dedicated-admins** グループを作成します。このグループには、クラスターまたは個別のプロジェクトレベルのロールが付与されます。ユーザーはこのグループに割り当てることができ、グループメンバーシップは OpenShift Dedicated 管理者アクセスを持つユーザーを定義します。ただし、設計上、サービスアカウントを通常のグループに追加することはできません。

その代わりに、**dedicated-admin** サービスはこの目的のために **dedicated-admin** という名前の特殊なプロジェクトを作成します。このプロジェクトのサービスアカウントグループには OpenShift Dedicated **admin** ロールが付与されます。これにより、**dedicated-admin** プロジェクト内のすべての

サービスアカウントに対して OpenShift Dedicated 管理者アクセスが付与されます。その後、これらのサービスアカウントを使用して、OpenShift Dedicated の管理者アクセスを必要とするすべてのアクションを実行できます。

dedicated-admins グループのメンバーであるユーザーには **dedicated-admin** ロールのパーミッションが付与され、これらのユーザーには **dedicated-admin** プロジェクトへの **edit** アクセスがあります。これらのユーザーはこのプロジェクトでサービスアカウントを管理し、必要に応じて新規のアカウントを作成することができます。

dedicated-reader ロールを持つユーザーには **dedicated-reader** プロジェクトへの編集および表示アクセスと、他のプロジェクトへの表示専用アクセスが付与されます。

5.3. DEDICATED の管理者ロールについて

OpenShift Dedicated クラスターの Dedicated 管理者用のアカウントのパーミッションやユーザーが作成するすべてのプロジェクトへのアクセスが拡大されています。

アカウントに **dedicated-admins-cluster** 認可ロールがバインドされると、クラスター内のユーザーが作成する新規プロジェクトの **dedicated-admins-project** ロールに自動的にバインドされます。

テンプレートなどの一連のリソース名で操作するには、 **create** などの動詞のセットに関連付けられたアクションを実行できます。それらのロールおよびそれらの動詞およびリソースのセットの詳細を表示するには、以下のコマンドを実行します。

```
$ oc describe clusterrole/dedicated-admins-cluster
$ oc describe clusterrole/dedicated-admins-project
```

動詞名のすべてが必ずしも **oc** コマンドに直接マップされる訳ではなく、通常これらの名前は実行できる CLI 操作のタイプに対応します。たとえば、**list** 動詞があると、**oc get** コマンドを実行して指定されたリソース名のすべてのオブジェクトの一覧を表示できますが、**get** 動詞の場合は、特定のオブジェクトの名前を知っている場合に **oc describe** コマンドを実行してその特定のオブジェクトの詳細を表示できます。

OpenShift Dedicated 管理者は、ユーザーに **dedicated-reader** ロールを付与できます。これにより、クラスターレベルでの表示専用アクセスと、すべてのユーザープロジェクトの表示アクセスが提供されます。

5.4. 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> (1)

serviceaccount "robot" created
```

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

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
```

5.5. 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6章 アプリケーションでのサービスアカウントの使用

6.1. サービスアカウントの概要

サービスアカウントは、コンポーネントが API に直接アクセスできるようにする OpenShift Dedicated 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 Dedicated 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。
- コンテナー内のアプリケーションが検出目的で API 呼び出しを実行する。
- 外部アプリケーションがモニターまたは統合目的で API 呼び出しを実行する。

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

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

すべてのサービスアカウントは以下の 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。

6.2. デフォルトのサービスアカウント

OpenShift Dedicated 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。

6.2.1. 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 Dedicated インフラストラクチャープロジェクト (**openshift-infra**) に作成され、クラスター全体での以下のロールが付与されます。

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

6.2.2. デフォルトのプロジェクトサービスアカウントおよびロール

3 つのサービスアカウントが各プロジェクトで自動的に作成されます。

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

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

6.3. サービスアカウントの作成

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

手順

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

```
$ oc get sa

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

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

```
$ oc create sa <service_account_name> 1

serviceaccount "robot" created
```

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

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
```

6.4. 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の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>
```

例:

```
$ 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7章 サービスアカウントの OAUTH クライアントとしての使用

7.1. 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。

7.1.1. 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す。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8章 スコープトークン

8.1. 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。

8.1.1. 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。

8.1.2. ロールスコープ

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

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

注記

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

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