Access control

Read more to learn about role-based access control and authentication.
Read more to learn about role-based access control and authentication.
Abstract

Read more to learn about role-based access control and authentication.
Red Hat Advanced Cluster Management for Kubernetes 2.5 Access control
CHAPTER 1. ACCESS CONTROL

Access control might need to manually be created and managed. You must configure authentication service requirements for Red Hat Advanced Cluster Management for Kubernetes to onboard workloads to Identity and Access Management (IAM). For more information see, Understanding authentication in the OpenShift Container Platform documentation.

Role-based access control and authentication identifies the user associated roles and cluster credentials. See the following files for information about access and credentials.

**Required access:** Cluster administrator

- Role-based access control

1.1. ROLE-BASED ACCESS CONTROL

Red Hat Advanced Cluster Management for Kubernetes supports role-based access control (RBAC). Your role determines the actions that you can perform. RBAC is based on the authorization mechanisms in Kubernetes, similar to Red Hat OpenShift Container Platform. For more information about RBAC, see the OpenShift RBAC overview in the OpenShift Container Platform documentation.

**Note:** Action buttons are disabled from the console if the user-role access is impermissible.

View the following sections for details of supported RBAC by component:

- Overview of roles
- RBAC implementation
- Cluster lifecycle RBAC
- Application lifecycle RBAC
- Governance lifecycle RBAC
- Observability RBAC

1.1.1. Overview of roles

Some product resources are cluster-wide and some are namespace-scoped. You must apply cluster role bindings and namespace role bindings to your users for consistent access controls. View the table list of the following role definitions that are supported in Red Hat Advanced Cluster Management for Kubernetes:

<table>
<thead>
<tr>
<th>Role</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>cluster-admin</strong></td>
<td>This is an OpenShift Container Platform default role. A user with cluster binding to the <strong>cluster-admin</strong> role is an OpenShift Container Platform super user, who has all access.</td>
</tr>
<tr>
<td>Role</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>open-cluster-management:cluster-manager-admin</td>
<td>A user with cluster binding to the open-cluster-management:cluster-manager-admin role is a Red Hat Advanced Cluster Management for Kubernetes super user, who has all access. This role allows the user to create a ManagedCluster resource.</td>
</tr>
<tr>
<td>open-cluster-management:admin:&lt;managed_cluster_name&gt;</td>
<td>A user with cluster binding to the open-cluster-management:admin:&lt;managed_cluster_name&gt; role has administrator access to the ManagedCluster resource named, &lt;managed_cluster_name&gt;. When a user has a managed cluster, this role is automatically created.</td>
</tr>
<tr>
<td>open-cluster-management:view:&lt;managed_cluster_name&gt;</td>
<td>A user with cluster binding to the open-cluster-management:view:&lt;managed_cluster_name&gt; role has view access to the ManagedCluster resource named, &lt;managed_cluster_name&gt;.</td>
</tr>
<tr>
<td>open-cluster-management:managedclusterset:admin:&lt;managed_clusterset_name&gt;</td>
<td>A user with cluster binding to the open-cluster-management:managedclusterset:admin:&lt;managed_clusterset_name&gt; role has administrator access to ManagedCluster resource named &lt;managed_clusterset_name&gt;. The user also has administrator access to managedcluster.cluster.open-cluster-management.io, clusterclaim.hive.openshift.io, clusterdeployment.hive.openshift.io, and clusterpool.hive.openshift.io resources, which has the managed cluster set labels: cluster.open-cluster-management.io and clusterset=&lt;managed_clusterset_name&gt;. A role binding is automatically generated when you are using a cluster set. See Creating and managing ManagedClusterSets to learn how to manage the resource.</td>
</tr>
<tr>
<td>open-cluster-management:managedclusterset:view:&lt;managed_clusterset_name&gt;</td>
<td>A user with cluster binding to the open-cluster-management:managedclusterset:view:&lt;managed_clusterset_name&gt; role has view access to the ManagedCluster resource named, &lt;managed_clusterset_name&gt;’. The user also has view access to managedcluster.cluster.open-cluster-management.io, clusterclaim.hive.openshift.io, clusterdeployment.hive.openshift.io, and clusterpool.hive.openshift.io resources, which has the managed cluster set labels: cluster.open-cluster-management.io, clusterset=&lt;managed_clusterset_name&gt;. For more details on how to manage managed cluster set resources, see Creating and managing ManagedClusterSets.</td>
</tr>
</tbody>
</table>
A user with the open-cluster-management:subscription-admin role can create Git subscriptions that deploy resources to multiple namespaces. The resources are specified in Kubernetes resource YAML files in the subscribed Git repository. Note: When a non-subscription-admin user creates a subscription, all resources are deployed into the subscription namespace regardless of specified namespaces in the resources. For more information, see the Application lifecycle RBAC section.

Admin, edit, view are OpenShift Container Platform default roles. A user with a namespace-scoped binding to these roles has access to open-cluster-management resources in a specific namespace, while cluster-wide binding to the same roles gives access to all of the open-cluster-management resources cluster-wide.

Important:

- Any user can create projects from OpenShift Container Platform, which gives administrator role permissions for the namespace.

- If a user does not have role access to a cluster, the cluster name is not visible. The cluster name is displayed with the following symbol: «.

1.1.2. RBAC implementation

RBAC is validated at the console level and at the API level. Actions in the console can be enabled or disabled based on user access role permissions. View the following sections for more information on RBAC for specific lifecycles in the product.

1.1.2.1. Cluster lifecycle RBAC

View the following cluster lifecycle RBAC operations.

- To create and administer all managed clusters:

  - Create a cluster role binding to the cluster role open-cluster-management:cluster-manager-admin by entering the following command:

    ```
    oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:cluster-manager-admin
    ```

    This role is a super user, which has access to all resources and actions. You can create cluster-scoped managedcluster resources, the namespace for the resources that manage the managed cluster, and the resources in the namespace with this role. You can also access provider connections and bare metal assets that are used to create managed clusters with this role.

- To administer a managed cluster named cluster-name:
Create a cluster role binding to the cluster role `open-cluster-management:admin:<cluster-name>` by entering the following command:

```
oc create clusterrolebinding (role-binding-name) --clusterrole=open-cluster-management:admin:<cluster-name>
```

This role has read and write access to the cluster-scoped `managedcluster` resource. This is needed because the `managedcluster` is a cluster-scoped resource and not a namespace-scoped resource.

Create a namespace role binding to the cluster role `admin` by entering the following command:

```
oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=admin
```

This role has read and write access to the resources in the namespace of the managed cluster.

To view a managed cluster named `cluster-name`:

Create a cluster role binding to the cluster role `open-cluster-management:view:<cluster-name>` by entering the following command:

```
oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:view:<cluster-name>
```

This role has read access to the cluster-scoped `managedcluster` resource. This is needed because the `managedcluster` is a cluster-scoped resource and not a namespace-scoped resource.

Create a namespace role binding to the cluster role `view` by entering the following command:

```
oc create rolebinding <role-binding-name> -n <cluster-name> --clusterrole=view
```

This role has read-only access to the resources in the namespace of the managed cluster.

View a list of the managed clusters that you can access by entering the following command:

```
oc get managedclusters.clusterview.open-cluster-management.io
```

This command is used by administrators and users without cluster administrator privileges.

View a list of the managed cluster sets that you can access by entering the following command:

```
oc get managedclustersets.clusterview.open-cluster-management.io
```

This command is used by administrators and users without cluster administrator privileges.

### 1.1.2.1.1. Cluster pools RBAC

View the following cluster pool RBAC operations.

To use cluster pool provision clusters:
As a cluster administrator, create a managed cluster set and grant administrator permission to roles by adding the role to the group.

- Grant `admin` permission to the `server-foundation-clusterset` managed cluster set with the following command:

  ```bash
  oc adm policy add-cluster-role-to-group open-cluster-management:clusterset-admin:server-foundation-clusterset
  server-foundation-team-admin
  ```

- Grant `view` permission to the `server-foundation-clusterset` managed cluster set with the following command:

  ```bash
  oc adm policy add-cluster-role-to-group open-cluster-management:clusterset-view:server-foundation-clusterset
  server-foundation-team-user
  ```

- Create a namespace for the cluster pool, `server-foundation-clusterpool`.

  - Grant `admin` permission to `server-foundation-clusterpool` for the `server-foundation-team-admin` by running the following commands:

    ```bash
    oc adm new-project server-foundation-clusterpool
    oc adm policy add-role-to-group admin server-foundation-team-admin --namespace
    server-foundation-clusterpool
    ```

- As a team administrator, create a cluster pool named `ocp46-aws-clusterpool` with a cluster set label, `cluster.open-cluster-management.io/clusterset=server-foundation-clusterset` in the cluster pool namespace.

  - The `server-foundation-webhook` checks if the cluster pool has the cluster set label, and if the user has permission to create cluster pools in the cluster set.

  - The `server-foundation-controller` grants `view` permission to the `server-foundation-clusterpool` namespace for `server-foundation-team-user`.

  - When a cluster pool is created, the cluster pool creates a `clusterdeployment`.

    - The `server-foundation-controller` grants `admin` permission to the `clusterdeployment` namespace for `server-foundation-team-admin`.

    - The `server-foundation-controller` grants `view` permission `clusterdeployment` namespace for `server-foundation-team-user`.

      Note: As a `team-admin` and `team-user`, you have `admin` permission to the `clusterpool`, `clusterdeployment`, and `clusterclaim`.

View the following console and API RBAC tables for cluster lifecycle:

**Table 1.2. Console RBAC table for cluster lifecycle**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clusters</td>
<td>read, update, delete</td>
<td>-</td>
<td>read</td>
</tr>
<tr>
<td>Resource</td>
<td>Admin</td>
<td>Edit</td>
<td>View</td>
</tr>
<tr>
<td>------------------------</td>
<td>--------------------------------</td>
<td>-------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Cluster sets</td>
<td>get, update, bind, join</td>
<td>edit role not mentioned</td>
<td>get</td>
</tr>
<tr>
<td>Managed clusters</td>
<td>read, update, delete</td>
<td>no edit role mentioned</td>
<td>get</td>
</tr>
<tr>
<td>Provider connections</td>
<td>create, read, update, and delete</td>
<td>-</td>
<td>read</td>
</tr>
<tr>
<td>Bare metal asset</td>
<td>create, read, update, delete</td>
<td>-</td>
<td>read</td>
</tr>
</tbody>
</table>

**Table 1.3. API RBAC table for cluster lifecycle**

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedclusters.cluster.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
</tbody>
</table>

*You can use `mcl` (singular) or `mcls` (plural) in commands for this API.*

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedclusters.view.open-cluster-management.io</td>
<td>read</td>
<td>read</td>
<td>read</td>
</tr>
</tbody>
</table>

*You can use `mcv` (singular) or `mcvs` (plural) in commands for this API.*

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedclusters.register.open-cluster-management.io/accept</td>
<td>update</td>
<td>update</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedclustersets.cluster.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
</tbody>
</table>

*You can use `mclset` (singular) or `mclsets` (plural) in commands for this API.*

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>managedclustersets.view.open-cluster-management.io</td>
<td>read</td>
<td>read</td>
<td>read</td>
</tr>
<tr>
<td>API</td>
<td>Admin</td>
<td>Edit</td>
<td>View</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-----------------------------</td>
<td>---------------</td>
<td>------</td>
</tr>
<tr>
<td>managedclustersetbinding.clust</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>You can use <strong>mclsetbinding</strong></td>
<td>(singular) or <strong>mclsetbindings</strong> (plural) in commands for this API.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>baremetalassets.inventory.clu</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>klusterletaddonconfigs.agent.cl</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>managedclusteractions.action.cl</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>managedclusterviews.view.cl</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>managedclusterinfos.internal.cl</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>manifestworks.work.open-clust</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>submarinerconfigs.submarineraddon.open-cluster-management.io</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>placements.cluster.open-cluster-management.io</td>
<td>create, read, update, del</td>
<td>read, update</td>
<td>read</td>
</tr>
</tbody>
</table>

1.1.2.2. Credentials role-based access control

The access to credentials is controlled by Kubernetes. Credentials are stored and secured as Kubernetes secrets. The following permissions apply to accessing secrets in Red Hat Advanced Cluster Management for Kubernetes:

- Users with access to create secrets in a namespace can create credentials.
- Users with access to read secrets in a namespace can also view credentials.
- Users with the Kubernetes cluster roles of admin and edit can create and edit secrets.
- Users with the Kubernetes cluster role of view cannot view secrets because reading the contents of secrets enables access to service account credentials.

1.1.2.3. Application lifecycle RBAC

When you create an application, the subscription namespace is created and the configuration map is created in the subscription namespace. You must also have access to the channel namespace. When you want to apply a subscription, you must be a subscription administrator. For more information on managing applications, see Creating an allow and deny list as subscription administrator.

View the following application lifecycle RBAC operations:

- To create and administer application on all managed clusters with a user named username:
  - Create a cluster role binding to the open-cluster-management:cluster-manager-admin cluster role and bind it to username, run the following command:
    ```bash
    oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:cluster-manager-admin --user=<username>
    ```
    This role is a super user, which has access to all resources and actions. You can create the namespace for the application and all application resources in the namespace with this role.

  - Option: You can create applications that deploy resources to multiple namespaces:
    - Create a cluster role binding to the open-cluster-management:subscription-admin cluster role, and bind it to a user named username. Run the following command:
      ```bash
      oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:subscription-admin --user=<username>
      ```

- To create and administer an application named application-name in the cluster-name managed cluster, with username user:
  - Create a cluster role binding to the open-cluster-management:admin:cluster role and bind it to username by entering the following command:
    ```bash
    oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:admin:<cluster-name> --user=<username>
    ```
    This role has read and write access to all application resources on the managed cluster, cluster-name. Repeat this if access for other managed clusters is required.

  - Create a namespace role binding to the application namespace using the admin role and bind it to username by entering the following command:
    ```bash
    oc create rolerolebinding <role-binding-name> -n <application-namespace> --clusterrole=admin --user=<username>
    ```
    This role has read and write access to all application resources in the application namespace. Repeat this if access for other applications is required or if the application deploys to multiple namespaces.
Option: You can create applications that deploy resources to multiple namespaces:

- Create a cluster role binding to the open-cluster-management:subscription-admin cluster role and bind it to `username` by entering the following command:

  ```
  oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:subscription-admin --user=<username>
  ```

- To view an application on a managed cluster named `cluster-name` with the user named `username`:

  - Create a cluster role binding to the `open-cluster-management:view` cluster role and bind it to `username` by entering the following command:

  ```
  oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:view:cluster-name --user=<username>
  ```

  This role has read access to all application resources on the managed cluster, `cluster-name`. Repeat this if access for other managed clusters is required.

  - Create a namespace role binding to the application namespace using the `view` role and bind it to `username`. Enter the following command:

  ```
  oc create rolebinding <role-binding-name> -n <application-namespace> --clusterrole=view --user=<username>
  ```

  This role has read access to all application resources in the application namespace. Repeat this if access for other applications is required.

View the following console and API RBAC tables for Application lifecycle:

**Table 1.4. Console RBAC table for application lifecycle**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>Channel</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>Subscription</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>Placement rule</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
</tbody>
</table>

**Table 1.5. API RBAC table for application lifecycle**

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Admin</td>
<td>Edit</td>
<td>View</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>applications.app.k8s.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>channels.apps.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>deployables.apps.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>helmreleases.apps.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>placementrules.apps.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>subscriptions.apps.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>configmaps</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>secrets</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
<tr>
<td>namespaces</td>
<td>create, read, update, delete</td>
<td>create, read, update, delete</td>
<td>read</td>
</tr>
</tbody>
</table>

### 1.1.2.4. Governance lifecycle RBAC

When a policy is created, the policy is created in the cluster. Roles for the governance lifecycle are namespace-scoped. A user must also have access to the managed cluster.

To perform governance lifecycle operations, users must have access to the namespace where the policy is created, along with access to the managed cluster where the policy is applied.

View the following examples:

- To create a policy in the `policy` namespace and apply it in a managed cluster named `cluster-name`:
  - Create a namespace role binding to the `policy` namespace using the `open-cluster-management:admin:` role. Run the following command:
    ```bash
    oc create rolebinding <role-binding-name> -n <policy-namespace> --clusterrole=admin --user=<username>
    ```

- To view a policy in a managed cluster:
- Create a cluster role binding to `open-cluster-management:admin` cluster role and bind it to the `view` role with the following command:

```bash
oc create clusterrolebinding <role-binding-name> --clusterrole=open-cluster-management:view:<cluster-name> --user=<username>
```

View the following console and API RBAC tables for governance lifecycle:

**Table 1.6. Console RBAC table for governance lifecycle**

<table>
<thead>
<tr>
<th>Resource</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>Policies</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>PlacementBindings</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>PlacementRules</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>PolicyAutomations</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
</tbody>
</table>

**Table 1.7. API RBAC table for governance lifecycle**

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>policies.policy.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>placementbindings.policy.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td>policyautomations.policy.open-cluster-management.io</td>
<td>create, read, update, delete</td>
<td>read, update</td>
<td>read</td>
</tr>
</tbody>
</table>

**1.1.2.5. Observability RBAC**

To view the observability metrics for a managed cluster, you must have `view` access to that managed cluster on the hub cluster. View the following list of observability features:

- Access managed cluster metrics. Users are denied access to managed cluster metrics, if they are not assigned to the `view` role for the managed cluster on the hub cluster.
- Search for resources.
To view observability data in Grafana, you must have a **RoleBinding** resource in the same namespace of the managed cluster. View the following **RoleBinding** example:

```yaml
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: <replace-with-name-of-rolebinding>
  namespace: <replace-with-name-of-managedcluster-namespace>
subjects:
  - kind: <replace with User|Group|ServiceAccount>
    apiGroup: rbac.authorization.k8s.io
    name: <replace with name of User|Group|ServiceAccount>
    roleRef:
      apiGroup: rbac.authorization.k8s.io
      kind: ClusterRole
      name: view
```

See [Role binding policy](#) for more information. See [Customizing observability](#) to configure observability.

To manage components of observability, view the following API RBAC table:

### Table 1.8. API RBAC table for observability

<table>
<thead>
<tr>
<th>API</th>
<th>Admin</th>
<th>Edit</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>multiclusterobservabilities.observability.open-cluster-management.io</code></td>
<td>create, read, update, and delete</td>
<td>read, update</td>
<td>read</td>
</tr>
<tr>
<td><code>searchcustomizations.search.open-cluster-management.io</code></td>
<td>create, get, list, watch, update, delete, patch</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td><code>policyreports.wgpolicyk8s.io</code></td>
<td>get, list, watch</td>
<td>get, list, watch</td>
<td>get, list, watch</td>
</tr>
</tbody>
</table>

To continue to learn more about securing your cluster, see [Risk and compliance](#).