Red Hat Advanced Cluster Management for Kubernetes 2.5

マルチクラスターエンジン

マルチクラスターエンジンオペレーターの使用方法

Last Updated: 2023-07-14

Red Hat Advanced Cluster Management for Kubernetes 2.5 マルチクラスターエンジン

マルチクラスターエンジンオペレーターの使用方法

法律上の通知

Copyright © 2023 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

マルチクラスターエンジンの使用方法をご覧ください。

目次

第1章 要件および推奨事項

Kubernetes Operator のマルチクラスターエンジンをインストールする前に、次のシステム設定要件と設定を確認してください。

- サポート対象のオペレーティングシステムおよびプラットフォーム
- ネットワーク設定

重要: 2.5 より前の Red Hat Advanced Cluster Management for Kubernetes がインストールされていないクラスターには、Kubernetes 用のマルチクラスターエンジンをインストールする必要があります。Kubernetes のマルチクラスターエンジンは、同じ管理コンポーネントの一部を提供するため、2.5 より前のバージョンでは Red Hat Advanced Cluster Management for Kubernetes と共存できません。これまで Red Hat をインストールしたことがないクラスターに Kubernetes 用のマルチクラスターエンジンをインストールすることをお勧めします。バージョン 2.5.0 以降で Red Hat Advanced Cluster Management for Kubernetes を使用している場合、Kubernetes 用のマルチクラスターエンジンは既にクラスターにインストールされています。

1.1. KUBERNETES クラスターおよびマネージドクラスターのマルチクラスターエンジンでサポートされているオペレーティングシステムとプラットフォーム

サポート対象のオペレーティングシステムについては、以下の表を参照してください。

プラットフォーム	Kubernetes クラスターのマルチクラスターエンジンでサポート	マネージドクラスターでサポート
Red Hat OpenShift Container Platform 3.11.200 および 3.11.x 以降のリリース	■	○
Red Hat OpenShift Container Platform 4.8.2 以降	○	○
Red Hat OpenShift Container Platform on Amazon Web Services	○	○
Red Hat OpenShift Container Platform on Microsoft Azure	○	○
Red Hat OpenShift Container Platform on Google Cloud Platform	○	○
Red Hat OpenShift Kubernetes Engine	■	○
Google Kubernetes Engine (Google GKE) (Kubernetes 1.17 以降)	■	○

Amazon Elastic Kubernetes Service (Amazon EKS) (Kubernetes 1.17.6 以降)	▮	○
Microsoft Azure Kubernetes Service (Microsoft AKS) (Kubernetes 1.19.6 以降)	▮	○

1.2. ネットワーク設定

以下のセクションで接続を許可するようにネットワークを設定します。

1.2.1. Kubernetes Operator クラスターネットワーク要件用のマルチクラスターエンジン

Kubernetes クラスターネットワーク要件のマルチクラスターエンジンについては、次の表を参照してください。

方向	接続	ポート (指定されている場合)
送信	クラウドプロバイダーの API	
Outbound	プロビジョニングしたマネージドクラスターの Kubernetes API サーバー	6443
送信および受信	マネージドクラスターの **WorkManager** サービスルート	443
受信	マネージドクラスターからの Kubernetes クラスター用のマルチクラスターエンジンの Kubernetes API サーバー	6443
受信	GitHub から Kubernetes クラスターのマルチクラスターエンジンへのコミット後のフック。この設定は、特定のアプリケーション管理機能を使用する場合にのみ必要です。	6443

1.2.2. マネージドクラスターネットワークの要件

マネージドクラスターネットワークの要件については、以下の表を参照してください。

方向	接続	ポート (指定されている場合)

方向	接続	ポート (指定されている場合)
送信および受信	Kubernetes クラスター用のマルチクラスターエンジンの Kubernetes API サーバー	6443

1.2.3. ホステッドコントロールプレーンのネットワーク要件 (テクノロジープレビュー)

ホステッドコントロールプレーンを使用する場合、**HypershiftDeployment** リソースには、次の表に示すエンドポイントへの接続が必要です。

方向	接続	ポート (指定されている場合)
Outbound	OpenShift Container Platform コントロールプレーンおよびワーカーノード	
Outbound	Amazon Web Services のホストされたクラスターのみ: AWS API および S3 API へのアウトバウンド接続	
Outbound	Microsoft Azure クラウドサービスのホストされたクラスターのみ: Azure API へのアウトバウンド接続	
Outbound	coreOS の ISO イメージと OpenShift Container Platform Pod のイメージレジストリーを格納する OpenShift Container Platform イメージリポジトリー	

第2章 スタートガイド

2.1. はじめに

Kubernetes Operator のマルチクラスターエンジンをインストールする前に、要件および推奨事項 でシステム設定の要件と設定を確認してください。クラスターにサポート対象の OpenShift Container Platform がインストールされ、稼働している場合には、ネットワーク接続時のオンラインインストール に進んでください。

2.2. クラスターの作成と管理

インストールすると、クラスターを作成、インポート、および管理する準備が整います。Kubernetes クラスター用のマルチクラスターエンジンから、管理する他の OpenShift Container Platform クラスターを作成できます。

1. 作成可能なマネージドクラスターの種類については、クラスターの作成 を参照してください。
2. 手動でインポートするクラスターがある場合は、クラスターのインポート を参照して、マネージドクラスターのインポート方法を確認してください。
3. クラスターを管理する必要がなくなったときに マネージドクラスターのデタッチ を表示できます。

2.3. MANIFESTWORK の例

プロセスと例は ManifestWork を使用したワークロードのデプロイ で確認できます。

第3章 ネットワーク接続時のオンラインインストール

Kubernetes オペレーター用のマルチクラスターエンジンは、Operator Lifecycle Manager とともにインストールされます。このマネージャーは、Kubernetes エンジン用のマルチクラスターエンジンを含むコンポーネントのインストール、アップグレード、および削除を管理します。

必要なアクセス権限: クラスターの管理者

重要:

- 2.5 より前の Red Hat Advanced Cluster Management for Kubernetes がインストールされていないクラスターに Kubernetes 用のマルチクラスターエンジンをインストールする必要があります。Kubernetes のマルチクラスターエンジンは、同じ管理コンポーネントの一部を提供するため、2.5 より前のバージョンでは Red Hat Advanced Cluster Management for Kubernetes と共存できません。これまで Red Hat をインストールしたことがないクラスターに Kubernetes 用のマルチクラスターエンジンをインストールすることをお勧めします。バージョン 2.5.0 以降で Red Hat Advanced Cluster Management for Kubernetes を使用している場合、Kubernetes 用のマルチクラスターエンジンは既にクラスターにインストールされています。

- OpenShift Container Platform 専用環境の場合は、**cluster-admin** 権限が必要です。デフォルトで、**dedicated-admin** ロールには OpenShift Container Platform Dedicated 環境で namespace を作成するために必要なパーミッションがありません。

- デフォルトでは、エンジンコンポーネントは追加設定なしで OpenShift Container Platform クラスターのワーカーノードにインストールされます。OpenShift Container Platform OperatorHub Web コンソールインターフェイスを使用するか、OpenShift Container Platform CLI を使用してエンジンをワーカーノードにインストールできます。

- OpenShift Container Platform クラスターをインフラストラクチャーノードで設定している場合は、追加のリソースパラメーターを使用して、OpenShift Container Platform CLI を使用してエンジンをそれらのインフラストラクチャーノードにインストールできます。すべてのエンジンコンポーネントがインフラストラクチャーノードをサポートしているわけではないため、インフラストラクチャーノードに Kubernetes 用のマルチクラスターエンジンをインストールする場合でも、一部のワーカーノードが必要です。詳細は、**インフラストラクチャーノードの Kubernetes エンジン用にマルチクラスターエンジンのインストール** セクションを参照してください。

- OpenShift Container Platform または Kubernetes のマルチクラスターエンジンによって作成されていない Kubernetes クラスターをインポートする場合は、イメージプルシークレットを設定する必要があります。イメージプルシークレットおよびその他の高度な設定方法については、このドキュメントの 詳細設定 セクションのオプションを参照してください。

  - 前提条件
  - OpenShift Container Platform インストールの確認
  - OperatorHub Web コンソールインターフェイスからのインストール
  - OpenShift Container Platform CLI からのインストール
  - インフラストラクチャーノードへのインストール

3.1. 前提条件

Kubernetes 用のマルチクラスターエンジンをインストールする前に、次の要件を確認してください。

- RedHat OpenShift Container Platform クラスターは、OpenShift Container Platform コンソールから OperatorHub カタログにある Kubernetes Operator のマルチクラスターエンジンにアクセスできるようにしている。
- catalog.redhat.com へのアクセスがある。
- お使いの環境に OpenShift Container Platform バージョン 4.8 以降をデプロイし、OpenShift Container Platform CLI でログインしている。以下の OpenShift Container Platform のインストールドキュメントを参照してください。
  - OpenShift Container Platform version 4.10
  - OpenShift Container Platform バージョン 4.9
  - OpenShift Container Platform バージョン 4.8
- OpenShift Container Platform のコマンドラインインターフェイス (CLI) は、**oc** コマンドを実行できるように設定している。Red Hat OpenShift CLI のインストールおよび設定の詳細は、CLI の使用方法 を参照してください。
- namespace の作成が可能な OpenShift Container Platform の権限を設定している。
- operator の依存関係にアクセスするには、インターネット接続が必要。
- OpenShift Container Platform Dedicated 環境にインストールするには、以下を参照してください。
  - OpenShift Container Platform Dedicated 環境が設定され、実行している。
  - エンジンのインストール先の OpenShift Container Platform Deplicated 環境での **cluster-admin** がある。

3.2. OPENSHIFT CONTAINER PLATFORM インストールの確認

レジストリー、ストレージサービスなど、サポート対象の OpenShift Container Platform バージョンがインストールされ、機能する状態である必要があります。OpenShift Container Platform のインストールの詳細は、OpenShift Container Platform のドキュメントを参照してください。

1. Kubernetes エンジンオペレーター用のマルチクラスターエンジンが OpenShift Container Platform クラスターにまだインストールされていないことを確認します。Kubernetes オペレーター用のマルチクラスターエンジンでは、OpenShift Container Platform クラスターごとに 1 つのインストールのみが許可されます。インストールがない場合は、次の手順に進みます。
2. OpenShift Container Platform クラスターが正しく設定されていることを確認するには、以下のコマンドを使用して OpenShift Container Platform Web コンソールにアクセスします。

   ```
   kubectl -n openshift-console get route
   ```

   以下の出力例を参照してください。

   ```
   openshift-console console console-openshift-console.apps.new-coral.purple-chesterfield.com
   console   https   reencrypt/Redirect     None
   ```

3. ブラウザーで URL を開き、結果を確認します。コンソール URL の表示が **console-openshift-console.router.default.svc.cluster.local** の場合は、Red Hat OpenShift Container Platform のインストール時に **openshift_master_default_subdomain** を設定します。**https://console-**

openshift-console.apps.new-coral.purple-chesterfield.com の例を参照してください。

Kubernetes 用のマルチクラスターエンジンのインストールに進むことができます。

3.3. OPERATORHUB WEB コンソールインターフェイスからのインストール

ベストプラクティス: OpenShift Container Platform ナビゲーションの **Administrator** ビューから、OpenShift Container Platform で提供される OperatorHub Web コンソールインターフェイスをインストールします。

1. **Operators** > **OperatorHub** を選択して利用可能な operator のリストにアクセスし、**multicluster engine for Kubernetes**operator を選択します。
2. **Install** をクリックします。
3. **Operator Installation** ページで、インストールのオプションを選択します。
   - Namespace:
     - Kubernetes エンジンのマルチクラスターエンジンは、独自の namespace またはプロジェクトにインストールする必要があります。
     - デフォルトでは、OperatorHub コンソールのインストールプロセスにより、**multicluster-engine** という名前の namespace が作成されます。**ベストプラクティス: multicluster-engine** namespace が使用可能な場合は、引き続き使用します。
     - **multicluster-engine** という名前の namespace がすでに存在する場合は、別の namespace を選択してください。
   - チャネル: インストールするリリースに対応するチャネルを選択します。チャネルを選択すると、指定のリリースがインストールされ、そのリリース内の今後のエラータ更新が取得されます。
   - 承認ストラテジー: 承認ストラテジーでは、サブスクライブ先のチャネルまたはリリースに更新を適用するのに必要な人の間のやり取りを特定します。
     - そのリリース内の更新が自動的に適用されるようにするには、デフォルトで選択されている **Automatic** を選択します。
     - **Manual** を選択して、更新が利用可能になると通知を受け取ります。更新がいつ適用されるかについて懸念がある場合は、これがベストプラクティスになる可能性があります。

   注記: 次のマイナーリリースにアップグレードするには、**OperatorHub** ページに戻り、最新リリースの新規チャネルを選択する必要があります。
4. **Install** を選択して変更を適用し、Operator を作成します。
5. **MultiClusterEngine** カスタムリソースを作成するには、次のプロセスを参照してください。
   a. OpenShift Container Platform コンソールナビゲーションで、**Installed Operators** > **multicluster engine for Kubernetes**を選択します。
   b. **MultiCluster Engine** タブを選択します。
   c. **Create MultiClusterEngine** を選択します。

d. YAML ファイルのデフォルト値を更新します。このドキュメントの **MultiClusterEngine advanced configuration** のオプションを参照してください。

- 次の例は、エディターにコピーできるデフォルトのテンプレートを示しています。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

6. **Create** を選択して、カスタムリソースを初期化します。Kubernetes エンジンのマルチクラスターエンジンがビルドされて起動するまで、最大 10 分かかる場合があります。**MultiClusterEngine** リソースが作成されると、リソースのステータスが **MultiCluster Engine** タブで **Available** になります。

3.4. OPENSHIFT CONTAINER PLATFORM CLI からのインストール

1. Operator 要件が含まれている Kubernetes エンジン namespace 用のマルチクラスターエンジンを作成します。次のコマンドを実行します。ここで、**namespace** は、Kubernetes エンジン namespace のマルチクラスターエンジンの名前です。**namespace** の値は、OpenShift Container Platform 環境では **プロジェクト** と呼ばれる場合があります。

```
oc create namespace <namespace>
```

2. プロジェクトの namespace を、作成した namespace に切り替えます。**namespace** は、手順 1 で作成した Kubernetes エンジン用のマルチクラスターエンジンの namespace の名前に置き換えてください。

```
oc project <namespace>
```

3. **OperatorGroup** リソースを設定するために YAML ファイルを作成します。namespace ごとに割り当てることができる Operator グループは 1 つだけです。**default** はお使いの operator グループ名に置き換えます。**namespace** はお使いのプロジェクトの namespace 名に置き換えます。以下の例を参照してください。

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: <default>
spec:
  targetNamespaces:
  - <namespace>
```

4. 以下のコマンドを実行して **OperatorGroup** リソースを作成します。**operator-group** は、作成した operator グループの YAML ファイル名に置き換えます。

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

5. OpenShift Container Platform サブスクリプションを設定するための YAML ファイルを作成します。ファイルは以下の例のようになります。

```
apiVersion: operators.coreos.com/v1alpha1
```

```
kind: Subscription
metadata:
  name: multicluster-engine
spec:
  sourceNamespace: openshift-marketplace
  source: redhat-operators
  channel: stable-1.0
  installPlanApproval: Automatic
  name: multicluster-engine
```

注記: インフラストラクチャーノードに Kubernetes エンジン用のマルチクラスターエンジンをインストールする場合は、Operator Lifecycle Manager サブスクリプションの追加設定 セクションを参照してください。

6. 以下のコマンドを実行して OpenShift Container Platform サブスクリプションを作成します。**subscription** は、作成したサブスクリプションファイル名に置き換えます。

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

7. YAML ファイルを作成して、**MultiClusterEngine** カスタムリソースを設定します。デフォルトのテンプレートは、以下の例のようになります。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec: {}
```

注記: インフラストラクチャーノードに Kubernetes エンジン用のマルチクラスターエンジンをインストールする方法については、MultiClusterEngine カスタムリソースの追加設定 セクションを参照してください。

8. 次のコマンドを実行して、**MultiClusterEngine** カスタムリソースを作成します。**custom-resource** は、カスタムリソースファイル名に置き換えます。

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

以下のエラーで、この手順に失敗した場合でも、リソースは作成され、適用されます。リソースが作成されてから数分後にもう一度コマンドを実行します。

```
error: unable to recognize "./mce.yaml": no matches for kind "MultiClusterEngine" in version
"operator.multicluster-engine.io/v1"
```

9. 以下のコマンドを実行してカスタムリソースを編集します。次のコマンドを実行した後、**MultiClusterEngine** カスタムリソースステータスが **status.phase** フィールドに **Available** として表示されるまでに最大 10 分かかる場合があります。

```
oc get mce -o=jsonpath='{.items[0].status.phase}'
```

Kubernetes operator 用のマルチクラスターエンジンを再インストールしても Pod が起動しない場合は、この問題を回避する手順について、再インストールに失敗する場合のトラブルシューティング を参照してください。

注記:

- **ClusterRoleBinding** を使用する **ServiceAccount** は、クラスター管理者特権を Kubernetes のマルチクラスターエンジンと、Kubernetes のマルチクラスターエンジンをインストールする namespace にアクセスできるすべてのユーザー認証情報に自動的に付与します。

3.5. インフラストラクチャーノードへのインストール

OpenShift Container Platform クラスターを、承認された管理コンポーネントを実行するためのインフラストラクチャーノードを組み込むように設定できます。インフラストラクチャーノードでコンポーネントを実行すると、それらの管理コンポーネントを実行しているノードの OpenShift Container Platform サブスクリプションクォータの割り当てる必要がなくなります。

OpenShift Container Platform クラスターにインフラストラクチャーノードを追加した後に、OpenShift Container Platform CLI からのインストール 手順に従い、以下の設定を Operator Lifecycle Manager サブスクリプションおよび **MultiClusterEngine** カスタムリソースに追加します。

3.5.1. インフラストラクチャーノードを OpenShift Container Platform クラスターに追加する

OpenShift Container Platform ドキュメントの インフラストラクチャーマシンセットの作成 で説明されている手順に従います。インフラストラクチャーノードは、Kubernetes の **taint** および **label** で設定され、管理以外のワークロードがそれらで稼働し続けます。

Kubernetes のマルチクラスターエンジンによって提供されるインフラストラクチャーノードの有効化と互換性を持たせるには、インフラストラクチャーノードに次の **taint** と **label** が適用されていることを確認してください。

```
metadata:
  labels:
    node-role.kubernetes.io/infra: ""
spec:
  taints:
  - effect: NoSchedule
    key: node-role.kubernetes.io/infra
```

3.5.2. Operator Lifecycle Manager サブスクリプションの追加設定

Operator Lifecycle Manager サブスクリプションを適用する前に、以下の追加設定を追加します。

```
spec:
  config:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - key: node-role.kubernetes.io/infra
      effect: NoSchedule
      operator: Exists
```

3.5.3. MultiClusterEngine カスタムリソースの追加設定

MultiClusterEngine カスタムリソースを適用する前に、以下の設定を追加します。

```
spec:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

第4章 ネットワーク切断状態でのインストール

インターネットに接続されていない Red Hat OpenShift Container Platform クラスターにマルチクラスターエンジン Operator をインストールする必要がある場合があります。ネットワーク接続のないエンジンにインストールする手順でも一部、オンラインインストールと同じ手順が必要になります。

重要: 2.5 より前の Red Hat Advanced Cluster Management for Kubernetes がインストールされていないクラスターには、Kubernetes 用のマルチクラスターエンジンをインストールする必要があります。Kubernetes のマルチクラスターエンジンは、同じ管理コンポーネントの一部を提供するため、2.5 より前のバージョンでは Red Hat Advanced Cluster Management for Kubernetes と共存できません。これまで Red Hat をインストールしたことがないクラスターに Kubernetes 用のマルチクラスターエンジンをインストールすることをお勧めします。バージョン 2.5.0 以降で Red Hat Advanced Cluster Management for Kubernetes を使用している場合、Kubernetes 用のマルチクラスターエンジンは既にクラスターにインストールされています。

インストール時にネットワークから直接パッケージにアクセスするのではなく、パッケージをダウンロードしておき、インストール時にアクセスできるようにする必要があります。

- 前提条件
- OpenShift Container Platform インストールの確認
- インフラストラクチャーノードでのハブクラスターのインストール準備

4.1. 前提条件

マルチクラスターエンジン Operator をインストールする前に、次の要件を満たしている必要があります。

- お使いの環境に Red Hat OpenShift Container Platform バージョン 4.8 以降をインストールし、コマンドラインインターフェイス (CLI) でログインしている。
- catalog.redhat.com へのアクセスがある。
  注記: ベアメタルクラスターを管理する場合は、Red Hat OpenShift Container Platform バージョン 4.8 以降が必要です。

  OpenShift Container Platform version 4.10、OpenShift Container Platform version 4.8 を参照してください。
- Red Hat OpenShift Container Platform の CLI バージョンは 4.8 以降を使用し、**oc** コマンドを実行できるように設定している。Red Hat OpenShift CLI のインストールおよび設定の詳細は、CLI の使用方法 を参照してください。
- namespace の作成が可能な Red Hat OpenShift Container Platform の権限を設定している。
- Operator の依存関係をダウンロードするために、インターネット接続のあるワークステーションが必要。

4.2. OPENSHIFT CONTAINER PLATFORM インストールの確認

- レジストリー、ストレージサービスなど、サポート対象の OpenShift Container Platform バージョンがクラスターにインストールされ、機能する状態である必要があります。OpenShift Container Platform バージョン 4.8 の詳細は、OpenShift Container Platform documentation を参照してください。

- 接続されている場合には、OpenShift Container Platform クラスターが正しく設定されていることを確認できます。OpenShift Container Platform Web コンソールにアクセスします。**kubectl -n openshift-console get route** コマンドを実行して、OpenShift Container Platform の Web コンソールにアクセスします。以下の出力例を参照してください。

```
openshift-console  console  console-openshift-console.apps.new-coral.purple-chesterfield.com  console  https  reencrypt/Redirect  None
```

この例のコンソール URL は **https:// console-openshift-console.apps.new-coral.purple-chesterfield.com** です。ブラウザーで URL を開き、結果を確認します。

コンソール URL の表示が **console-openshift-console.router.default.svc.cluster.local** の場合は、Red Hat OpenShift Container Platform のインストール時に **openshift_master_default_subdomain** を設定します。

4.3. 非接続環境でのインストール

重要: 必要なイメージをミラーリングレジストリーにダウンロードし、非接続環境で Operator をインストールする必要があります。ダウンロードがないと、デプロイメント時に **ImagePullBackOff** エラーが表示される可能性があります。

次の手順に従って、切断された環境に Kubernetes Operator 用のマルチクラスターエンジンをインストールします。

1. ミラーレジストリーを作成します。ミラーレジストリーがない場合は、Red Hat OpenShift Container Platform ドキュメントの 非接続インストールのイメージのミラーリング の手順を実行してミラーレジストリーを作成してください。
ミラーレジストリーがすでにある場合は、既存のレジストリーを設定して使用できます。

2. **注記:** ベアメタルの場合のみ、**install-config.yaml** ファイルに、接続なしのレジストリーの証明書情報を指定する必要があります。保護された切断されたレジストリー内のイメージにアクセスするには、Kubernetes Operator のマルチクラスターエンジンがレジストリーにアクセスできるように、証明書情報を提供する必要があります。

   a. レジストリーから証明書情報をコピーします。

   b. エディターで **install-config.yaml** ファイルを開きます。

   c. **additionalTrustBundle: |** のエントリーを検索します。

   d. **additionalTrustBundle** の行の後に証明書情報を追加します。追加後の内容は以下の例のようになります。

```
additionalTrustBundle: |
  -----BEGIN CERTIFICATE-----
  certificate_content
  -----END CERTIFICATE-----
sshKey: >-
```

3. **重要:** 以下のガバナンスポリシーが必要な場合は、非接続イメージレジストリーの追加ミラーが必要です。

   - Container Security Operator ポリシー: イメージはソース **registry.redhat.io/quay** にあります。

- Compliance Operator ポリシー: イメージはソース **registry.redhat.io/compliance** にあります。
- Gatekeeper Operator ポリシー: イメージはソース **registry.redhat.io/rhacm2** にあります。
  3 つのすべての Operator については、以下のミラー一覧を参照してください。

```
- mirrors:
  - <your_registry>/rhacm2
  source: registry.redhat.io/rhacm2
- mirrors:
  - <your_registry>/quay
  source: registry.redhat.io/quay
- mirrors:
  - <your_registry>/compliance
  source: registry.redhat.io/compliance
```

4. **install-config.yaml** ファイルを保存します。
5. **rhacm-policy.yaml** という名前の **ImageContentSourcePolicy** を含めて yaml ファイルを作成します。**注記:** 実行中のクラスターでこれを変更すると、すべてのノードのローリング再起動が実行されます。

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: mce-repo
spec:
  repositoryDigestMirrors:
  - mirrors:
    - mirror.registry.com:5000/multicluster-engine
    source: registry.redhat.io/multicluster-engine
```

6. 以下のコマンドを入力して ImageContentSourcePolicy ファイルを適用します。

```
oc apply -f mce-policy.yaml
```

7. ネットワーク接続されていない Operator Lifecycle Manager の Red Hat Operator と コミュニティーの Operator を有効にします。
   Kubernetes Operator 用のマルチクラスターエンジンは、Operator Lifecycle Manager Red Hat カタログに含まれています。
8. Red Hat Operator カタログの非接続 Operator Lifecycle Manager を設定します。Red Hat OpenShift Container Platform ドキュメントの ネットワークが制限された環境での Operator Lifecycle Manager の使用 の手順を実行します。
9. 切断された Operator Lifecycle Manager にイメージが作成されたので、引き続き、Operator Lifecycle Manager カタログから Kubernetes 用の Kubernetes オペレーター用のマルチクラスターエンジンをインストールします。

必要な手順については、ネットワーク接続時のオンラインインストール の手順を参照してください。

第5章 コンソールの概要

OpenShift Container Platform コンソールプラグインは OpenShift Container Platform 4.10 Web コンソールで利用可能であり、統合することができます。この機能を使用するには、コンソールプラグインを有効にしておく必要があります。マルチクラスターエンジンの Operator は、**Infrastructure** および **Credentials** のナビゲーション項目から特定のコンソール機能を表示します。Red Hat Advanced Cluster Management をインストールすると、より多くのコンソール機能が表示されます。

注記: プラグインが有効になっている OpenShift Container Platform 4.10 の場合は、ドロップダウンメニューから **All Clusters** を選択することにより、クラススイッチャーから OpenShift Container Platform コンソール内で Red Hat Advanced Cluster Management にアクセスできます。

1. プラグインを無効にするには、OpenShift Container Platform コンソールの **Administrator** パースペクティブにいることを確認してください。

2. ナビゲーションで **Administration** を探し、**Cluster Settings** をクリックし、続いて **Configuration** タブをクリックします。

3. **Configuration resources** のリストから、**operator.openshift.io** API グループが含まれる **Console** リソースをクリックします。この API グループには、Web コンソールのクラスター全体の設定が含まれています。

4. **Console plug-ins** タブをクリックします。**mce** プラグインが一覧表示されます。**注記:** Red Hat Advanced Cluster Management がインストールされている場合は、**acm** としても表示されます。

5. テーブルからプラグインのステータスを変更します。しばらくすると、コンソールを更新するように求められます。

第6章 詳細設定

Kubernetes Operator のマルチクラスターエンジンは、必要なすべてのコンポーネントをデプロイする Operator を使用してインストールされます。Kubernetes Operator のマルチクラスターエンジンは、インストール中またはインストール後に、**MultiClusterEngine** カスタムリソースに次の属性の 1 つ以上を追加することでさらに設定できます。

6.1. カスタムイメージプルシークレット

OpenShift Container Platform または Kubernetes Operator 用のマルチクラスターエンジンによって作成されていない Kubernetes クラスターをインポートする場合は、OpenShift Container Platform プルシークレット情報を含むシークレットを生成して、ディストリビューションレジストリーから資格のあるコンテンツにアクセスします。

OpenShift Container Platform クラスターのシークレット要件は、OpenShift Container Platform および multicluster engine for Kubernetes により自動で解決されるため、他のタイプの Kubernetes クラスターをインポートして管理しない場合には、このシークレットを作成する必要はありません。

重要: これらのシークレットは namespace に依存するため、エンジンに使用する namespace にいることを確認してください。

1. cloud.redhat.com/openshift/install/pull-secret から **Download pull secret** を選択して、OpenShift Container Platform のプルシークレットファイルをダウンロードします。OpenShift Container Platform プルシークレットは Red Hat カスタマーポータル ID に関連しており、すべての Kubernetes プロバイダーで同じです。

2. 以下のコマンドを実行してシークレットを作成します。

```
oc create secret generic <secret> -n <namespace> --from-file=.dockerconfigjson=<path-to-pull-secret> --type=kubernetes.io/dockerconfigjson
```

   - **secret** は作成するシークレット名に置き換えます。
   - シークレットは namespace 固有であるため、**namespace** はプロジェクトの namespace に置き換えます。
   - **path-to-pull-secret** はダウンロードした OpenShift Container Platform のプルシークレットへのパスに置き換えます。

以下の例では、カスタムプルシークレットを使用する場合に使用する **spec.imagePullSecret** テンプレートを表示しています。**secret** は、プルシークレット名に置き換えます。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  imagePullSecret: <secret>
```

6.2. ターゲット NAMESPACE

MultiClusterEngine カスタムリソースで場所を指定することにより、指定された namespace にオペランドをインストールできます。この namespace は、**MultiClusterEngine** カスタムリソースの適用時に作成されます。

重要: ターゲット namespace が指定されていない場合、Operator は **multicluster-engine** namespace にインストールし、**MultiClusterEngine** カスタムリソース仕様で設定します。

次の例は、ターゲット namespace を指定するために使用できる **spec.targetNamespace** テンプレートを示しています。**target** を宛先 namespace の名前に置き換えます。 **注記: target** namespace を **default** namespace にすることはできません。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  targetNamespace: <target>
```

6.3. AVAILABILITYCONFIG

Red Hat Advanced Cluster Management ハブクラスターには、**High** と **Basic** の 2 つのアイラビリティーがあります。デフォルトでは、ハブクラスターには **High** の可用性があります。これにより、ハブクラスターコンポーネントに **replicaCount 2** が提供されます。これにより、フェイルオーバー時のサポートが向上しますが、**Basic** 可用性よりも多くのリソースを消費します。これにより、コンポーネントには **replicaCount 1** が提供されます。

以下の例は、**Basic** の可用性のある **spec.availabilityConfig** テンプレートを示しています。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  availabilityConfig: "Basic"
```

6.4. NODESELECTOR

MultiClusterEngine でノードセレクターのセットを定義して、クラスター上の特定のノードにインストールできます。以下の例は、**node-role.kubernetes.io/infra** ラベルの付いたノードに Red Hat Advanced Cluster Management Pod を割り当てる **spec.nodeSelector** を示しています。

```
spec:
  nodeSelector:
    node-role.kubernetes.io/infra: ""
```

6.5. TOLERATIONS

許容範囲のリストを定義して、**MultiClusterEngine** がクラスターで定義された特定の taint を許容できるようにすることができます。以下の例は、**node-role.kubernetes.io/infra** Taint に一致する **spec.tolerations** を示しています。

```
spec:
  tolerations:
  - key: node-role.kubernetes.io/infra
    effect: NoSchedule
    operator: Exists
```

以前の infra-node Toleration は、設定に Toleration を指定せずにデフォルトで Pod に設定されます。設定で許容値をカスタマイズすると、このデフォルトの動作が置き換えられます。

6.6. MANAGEDSERVICEACCOUNT アドオン (テクノロジープレビュー)

デフォルトでは、**Managed-ServiceAccount** アドオンは無効になっています。このコンポーネントを有効にすると、マネージドクラスターでサービスアカウントを作成または削除できます。このアドオンを有効にしてインストールするには、**spec.overrides** の **MultiClusterEngine** 仕様に以下を含めます。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: managedserviceaccount-preview
      enabled: true
```

Managed-ServiceAccount アドオンは、**MultiClusterEngine** の作成後に、コマンドラインでリソースを編集し、**managedserviceaccount-preview** コンポーネントを **enabled: true** に設定することで有効にできます。または、次のコマンドを実行して、<multiclusterengine-name> を **MultiClusterEngine** リソースの名前に置き換えることもできます。

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path": "/spec/overrides/components/-","value":{"name":"managedserviceaccount-preview","enabled":true}}]'
```

有効にするには、ManagedServiceAccount アドオンの有効化 を参照してください。

6.7. HYPERSHIFT アドオン (テクノロジープレビュー)

デフォルトでは、**Hypershift** アドオンは無効になっています。このアドオンを有効にしてインストールするには、**spec.overrides** の **MultiClusterEngine** 値に以下を含めます。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: hypershift-preview
      enabled: true
```

Hypershift アドオンは、**MultiClusterEngine** の作成後に、コマンドラインでリソースを編集し、**hypershift-preview** コンポーネントを **enabled: true** に設定することで有効にできます。または、次のコマンドを実行して、<multiclusterengine-name> を **MultiClusterEngine** リソースの名前に置き換えることもできます。

```
oc patch MultiClusterEngine <multiclusterengine-name> --type=json -p='[{"op": "add", "path": "/spec/overrides/components/-","value":{"name":"hypershift-preview","enabled":true}}]'
```

第7章 MANAGEDSERVICEACCOUNT アドオンの有効化 (テクノロジープレビュー)

Kubernetes Operator 用のマルチクラスターエンジンをインストールすると、**ManagedServiceAccount** アドオンはデフォルトで無効になります。このコンポーネントを有効にすると、マネージドクラスターでサービスアカウントを作成または削除できます。

必要なアクセス権限: 編集

ManagedServiceAccount カスタムリソースがハブクラスターの **<managed_cluster>** namespace に作成されると、**ServiceAccount** がマネージドクラスターに作成されます。

TokenRequest は、マネージドクラスターの **ServiceAccount** を使用して、マネージドクラスターの Kubernetes API サーバーに対して行われます。トークンは、ハブクラスターの **<target_managed_cluster>** namespace の **Secret** に保存されます。

注記 トークンは期限切れになり、ローテーションされる可能性があります。トークンリクエストの詳細については、TokenRequest を参照してください。

7.1. 前提条件

- お使いの環境に Red Hat OpenShift Container Platform バージョン 4.9 以降をインストールし、コマンドラインインターフェイス (CLI) でログインしている。
- Kubernetes Operator 用のマルチクラスターエンジンがインストールされている。

7.2. MANAGEDSERVICEACCOUNT の有効化

ハブクラスターとマネージドクラスターの **Managed-ServiceAccount** アドオンを有効にするには、次の手順を実行します。

1. ハブクラスターで **ManagedServiceAccount** アドオンを有効にします。詳細は、詳細設定 を参照してください。
2. **ManagedServiceAccount** アドオンをデプロイし、それをターゲットのマネージドクラスターに適用します。次の YAML ファイルを作成し、**target_managed_cluster** を **Managed-ServiceAccount** アドオンを適用するマネージドクラスターの名前に置き換えます。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: managed-serviceaccount
  namespace: <target_managed_cluster>
spec:
  installNamespace: open-cluster-management-agent-addon
```

3. 次のコマンドを実行して、ファイルを適用します。

```
oc apply -f -
```

これで、マネージドクラスターの **Managed-ServiceAccount** プラグインが有効になりました。**ManagedServiceAccount** を設定するには、次の手順を参照してください。

4. 次の YAML ソースを使用して **ManagedServiceAccount** カスタムリソースを作成します。

```
apiVersion: authentication.open-cluster-management.io/v1alpha1
kind: ManagedServiceAccount
metadata:
  name: <managed_serviceaccount_name>
  namespace: <target_managed_cluster>
spec:
  rotation: {}
```

- **managed_serviceaccount_name** を **ManagedServiceAccount** の名前に置き換えます。
- **target_managed_cluster** を、**ManagedServiceAccount** を適用するマネージドクラスターの名前に置き換えます。

5. 確認するには、**ManagedServiceAccount** オブジェクトのステータスで **tokenSecretRef** 属性を表示して、シークレット名と namespace を見つけます。アカウントとクラスター名を使用して次のコマンドを実行します。

```
oc get managedserviceaccount <managed_serviceaccount_name> -n
<target_managed_cluster> -o yaml
```

6. マネージドクラスターで作成された **ServiceAccount** に接続されている取得されたトークンを含む **Secret** を表示します。以下のコマンドを実行します。

```
oc get secret <managed_serviceaccount_name> -n <target_managed_cluster> -o yaml
```

第8章 クラスターの作成

Kubernetes のマルチクラスターエンジンは、内部 Hive コンポーネントを使用して Red Hat OpenShift Container Platform クラスターを作成します。クラスターの作成方法については、以下の情報を参照してください。

- 前提条件
- ClusterDeployment を使用してクラスターを作成する
- クラスタープールを使用してクラスターを作成する

8.1. 前提条件

クラスターを作成する前に、clusterImageSets リポジトリーのクローンを作成し、ハブクラスターに適用する必要があります。以下の手順を参照してください。

1. 次のコマンドを実行してクローンを作成します。

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/release-2.5
```

2. 次のコマンドを実行して、ハブクラスターに適用します。

```
find clusterImageSets/fast -type d -exec oc apply -f {} \; 2> /dev/null
```

クラスターを作成するときに、Red Hat OpenShift Container Platform リリースイメージを選択します。

8.2. CLUSTERDEPLOYMENT を使用してクラスターを作成する

ClusterDeployment は、Hive カスタムリソースです。個々のクラスターを作成する方法については、次のドキュメントを参照してください。

Using Hive のドキュメントに従って、**ClusterDeployment** カスタムリソースを作成します。

8.3. CLUSTERPOOL を使用してクラスターを作成

ClusterPool は、複数のクラスターを作成するために使用される Hive カスタムリソースでもあります。**ClusterPool** API を使用してクラスターを作成します。

Cluster Pools のドキュメントに従って、クラスターをプロビジョニングします。

第9章 クラスターのインポート

Kubernetes Operator 用のマルチクラスターエンジンをインストールすると、クラスターをインポートして管理できるようになります。Red Hat OpenShift Container Platform CLI を使用すると、インポートするクラスターの **kubeconfig** ファイルを使用してクラスターをインポートできます。または、インポートするクラスターでインポートコマンドを手動で実行することもできます。どちらの手順も文書化されています。

CLI からインポートするには、次の手順を参照してください。

- 前提条件
- インポートの準備
- 自動インポートシークレットを使用したインポート
- 手動コマンドによるインポート
- マネージドクラスターのデタッチ

9.1. 前提条件

- Linux (x86_64、s390x、ppc64le) または macOS が使用可能である。
- Kubernetes Operator 用のマルチクラスターエンジンをインストールし、MultiClusterEngine カスタムリソースを Kubernetes クラスターにインストールしている。
- 管理予定の別のクラスターとインターネット接続が必要。
- **oc** コマンドを実行するために、OpenShift Container Platform の CLI バージョン 4.8 以降が必要。Red Hat OpenShift CLI (**oc**) のインストールおよび設定の詳細は、CLI の使用方法 を参照してください。
  注記: CLI ツールのインストールファイルを OpenShift Container Platform コンソールからダウンロードします。
- OpenShift Container Platform によって作成されていないクラスターをインポートする場合は、**multiclusterengine.spec.imagePullSecret** を定義している。このシークレットは、Kubernetes Operator 用のマルチクラスターエンジンがインストールされたときに作成された可能性があります。このシークレットを定義する方法の詳細については、カスタムイメージプルシークレット を参照してください。

9.2. インポートの準備

1. **engine cluster** にログインします。**エンジンクラスター** は、Kubernetes Operator 用のマルチクラスターエンジンとカスタムリソースを含むクラスターです。以下のコマンドを実行します。

   ```
   oc login
   ```

2. エンジンクラスターで次のコマンドを実行して、プロジェクトを作成します。
   注記: CLUSTER_NAME で定義したクラスター名は、**.yaml** ファイルおよびコマンドでクラスターの namespace としても使用します。

   ```
   oc new-project ${CLUSTER_NAME}
   ```

3. 以下のコマンドを実行して namespace を作成します。

```
oc label namespace ${CLUSTER_NAME} cluster.open-cluster-
management.io/managedCluster=${CLUSTER_NAME}
```

4. 以下の YAML 例のように、**ManagedCluster** の例を編集します。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: ${CLUSTER_NAME}
spec:
  hubAcceptsClient: true
```

5. **オプション:** このリリースでは、ハブクラスターを自動的にインポートして、**local-cluster** と呼ばれるマネージドクラスターにすることは **できません**。マネージドクラスターを手動で **local-cluster** にできるようにするには、**metadata.labels.local-cluster: "true"** を追加します。次の YAML の例を参照して、名前が **local-cluster** であることを確認してください。**local-cluster** が名前でない場合、インポートは失敗するか、予期しない結果が発生します。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    local-cluster: "true"
    cloud: auto-detect
    vendor: auto-detect
  name: local-cluster
spec:
  hubAcceptsClient: true
```

6. ファイルは **managed-cluster.yaml** として保存します。

7. 以下のコマンドを使用して、YAML ファイルを適用します。

```
oc apply -f managed-cluster.yaml
```

9.3. 自動インポートシークレットを使用したインポート

エンジンクラスターにログインしたまま、次の手順に進みます。

1. インポートするクラスターの **kubeconfig** ファイル、またはインポートするクラスターの kube API サーバーとトークンを取得します。**kubeconfig** ファイルまたは kube API サーバーおよびトークンの場所を特定する方法については、Kubernetes クラスターのドキュメントを参照してください。

2. **kubeconfig** またはサーバー/トークンのペアを使用して、次のテンプレートに類似したコンテンツを含む YAML ファイルを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: auto-import-secret
stringData:
```

```
  # the following value to specify the retry times when your cluster failed to import
  autoImportRetry: "5"
  # If you are using the kubeconfig file, add the following value for the kubeconfig file
  # that has the current context set to the cluster to import:
  kubeconfig: |- <kubeconfig_file>
  # If you are using the server/token pair, add the following two values:
  server: <cluster_api_url>
  token: <Token to access the cluster>
type: Opaque
```

3. ファイルを auto-import-secret.yaml として保存します。

4. インポートするクラスターの **kubeconfig** ファイルを使用して、**${CLUSTER_NAME}** namespace にインポートシークレットを生成します。**kubeconfig** および **CLUSTER_NAME** へのパスを指定して次のコマンドを実行します。

```
oc apply -f auto-import-secret.yaml
```

注記: 自動インポートシークレットは 1 回使用され、インポートプロセスの完了時に削除されます。

5. インポートしたクラスターのステータス (**JOINED** および **AVAILABLE**) を確認します。Kubernetes クラスターのマルチクラスターエンジンから次のコマンドを実行します。

```
oc get managedcluster ${CLUSTER_NAME}
```

インポートする別のクラスターで次の手順に進みます。

6. インポートするクラスターにログインします。以下のコマンドを実行します。

```
oc login
```

7. インポートするクラスターの Pod ステータスを検証します。以下のコマンドを実行します。

```
oc get pod -n open-cluster-management-agent
```

8. アドオンは、インポートするクラスターが使用 **AVAILABLE** になった後にインストールされます。クラスター上のアドオンの Pod ステータスを検証します。以下のコマンドを実行します。

```
oc get pod -n open-cluster-management-agent-addon
```

クラスターがインポートされました。

9.4. 手動コマンドでインポート

重要: import コマンドには、インポートした各クラスターにコピーされるプルシークレット情報が含まれます。インポートしたクラスターにアクセスできるユーザーであれば誰でも、プルシークレット情報を表示することもできます。

1. エンジンクラスターのインポートコントローラーによって生成された **klusterlet-crd.yaml** を取得します。以下のコマンドを実行します。

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.crds\\.yaml} | base64 --decode > klusterlet-crd.yaml
```

2. エンジンクラスターのインポートコントローラーによって生成された **import.yaml** を取得します。以下のコマンドを実行します。

```
oc get secret ${CLUSTER_NAME}-import -n ${CLUSTER_NAME} -o jsonpath=
{.data.import\\.yaml} | base64 --decode > import.yaml
```

インポートする別のクラスターで次の手順に進みます。

3. インポートするクラスターにログインします。

```
oc login
```

4. 前の手順で生成した **klusterlet-crd.yaml** を適用します。以下のコマンドを実行します。

```
oc apply -f klusterlet-crd.yaml
```

5. 以前に生成した **import.yaml** ファイルを適用します。以下のコマンドを実行します。

```
oc apply -f import.yaml
```

6. インポートするクラスターの Pod ステータスを検証します。以下のコマンドを実行します。

```
oc get pod -n open-cluster-management-agent
```

7. インポートしているクラスターの **JOINED** および **AVAILABLE** のステータスを検証します。エンジンクラスターから、次のコマンドを実行します。

```
oc get managedcluster ${CLUSTER_NAME}
```

アドオンは、インポートするクラスターの **AVAILABLE** の後にインストールされます。

8. インポートするクラスター上のアドオンの Pod ステータスを検証します。以下のコマンドを実行します。

```
oc get pod -n open-cluster-management-agent-addon
```

これでクラスターがインポートされ、エンジンクラスターからそのクラスターを管理できます。

9.5. マネージドクラスターのデタッチ

マネージドクラスターは、正常にインポートされたクラスターです。マネージドクラスターをエンジンクラスターからデタッチするには、次のコマンドを実行します。

```
oc delete managedcluster ${CLUSTER_NAME}
```

これで、クラスターが切り離されました。

第10章 MANIFESTWORK を使用したワークロードの展開

Kubernetes クラスターのマルチクラスターエンジンからマネージドクラスターにワークロードをデプロイできます。例:Kubernetes クラスターのマルチクラスターエンジンからマネージドクラスターに基本的なデプロイを作成するには、**ManifestWork** を使用した次のサンプルを参照してください。

1. Kubernetes クラスターのマルチクラスターエンジンにログインします。

```
oc login
```

2. 次の例のように、YAML ファイルを作成して **ManifestWork** リソースを設定します。**CLUSTER_NAME** を クラスターのインポート ドキュメントでインポートしたマネージドクラスターの名前に置き換えます。サンプルの YAML は、ファイルを適用すると、マネージドクラスターの **default** の namespace にデプロイされます。

```
apiVersion: work.open-cluster-management.io/v1
kind: ManifestWork
metadata:
  name: hello-work
  namespace: ${CLUSTER_NAME}
  labels:
    app: hello
spec:
  workload:
    manifests:
    - apiVersion: apps/v1
      kind: Deployment
      metadata:
        name: hello
        namespace: default
      spec:
        selector:
          matchLabels:
            app: hello
        template:
          metadata:
            labels:
              app: hello
          spec:
            containers:
              - name: hello
                image: quay.io/asmacdo/busybox
                command: ['/bin/sh', '-c', 'echo "Hello, Kubernetes!" && sleep 300']
    - apiVersion: v1
      kind: Service
      metadata:
        labels:
          app: hello
        name: hello
        namespace: default
      spec:
        ports:
        - port: 8000
          protocol: TCP
```

```
    targetPort: 8000
  selector:
    app: hello
```

3. YAML ファイルを適用します。以下のコマンドを実行します。

```
oc apply -f manifestwork.yaml
```

4. 次のコマンドを実行して、Kubernetes クラスターのマルチクラスターエンジンから **ManifestWork** のステータスを確認します。

```
oc get manifestwork -n ${CLUSTER_NAME} hello-work -o yaml
```

5. マネージドクラスターにログインして、結果を表示します。以下のコマンドを使用します。

```
oc login
```

6. Kubernetes クラスター用のマルチクラスターエンジンで作成したデプロイを表示します。

```
$ oc get deploy -n default
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
hello   1/1     1            1           37s
```

次のコマンドを使用して、作成された Pod を表示することもできます。

```
$ oc get pod
NAME                     READY   STATUS    RESTARTS   AGE
hello-65f58985ff-4rm57   1/1     Running   0          42s
```

作成された Pod のログを表示すると、次のようなメッセージが表示されます。

```
$ oc logs hello-65f58985ff-4rm57
Hello, Kubernetes!
```

第11章 API

クラスターライフサイクル管理のために、Kubernetes Operator のマルチクラスターエンジンの API にアクセスできます。**ユーザーに必要なアクセス権:** ロールが割り当てられているアクションのみを実行できます。詳細は、以下の各リソースに関する API のドキュメントを参照してください。

- Clusters API
- ClusterSets API (v1beta1)
- Clusterview API
- ClusterSetBindings API (v1beta1)
- MultiClusterEngine API
- Placements API (v1alpha1)
- PlacementDecisions API (v1alpha1)
- マネージドサービスアカウント (テクノロジープレビュー)

11.1. CLUSTERS API

11.1.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンのクラスターリソースを対象としています。クラスターリソースには、create、query、delete、update の 4 つの要求を使用できます。

11.1.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.1.1.2. タグ

- cluster.open-cluster-management.io: クラスターを作成して管理します。

11.1.2. パス

11.1.2.1. 全クラスターのクエリー

```
GET /cluster.open-cluster-management.io/v1/managedclusters
```

11.1.2.1.1. 説明

クラスターに対してクエリーを実行して詳細を確認します。

11.1.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.1.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.1.2.1.4. 消費

- **cluster/yaml**

11.1.2.1.5. タグ

- cluster.open-cluster-management.io

11.1.2.2. クラスターの作成

```
POST /cluster.open-cluster-management.io/v1/managedclusters
```

11.1.2.2.1. 説明

クラスターの作成

11.1.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

型	名前	説明	スキーマ
ボディー	body 必須	作成するクラスターを記述するパラメーター	クラスター

11.1.2.2.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.1.2.2.4. 消費

- **cluster/yaml**

11.1.2.2.5. タグ

- cluster.open-cluster-management.io

11.1.2.2.6. HTTP 要求の例

11.1.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedCluster",
  "metadata" : {
    "labels" : {
      "vendor" : "OpenShift"
    },
    "name" : "cluster1"
  },
  "spec": {
    "hubAcceptsClient": true,
    "managedClusterClientConfigs": [
      {
        "caBundle": "test",
        "url": "https://test.com"
      }
    ]
```

```
  },
  "status" : { }
}
```

11.1.2.3. 単一クラスターのクエリー

```
GET /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}
```

11.1.2.3.1. 説明

1 つのクラスターに対してクエリーを実行して詳細を確認します。

11.1.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**cluster_name** **必須**	問い合わせるクラスターの名前。	文字列

11.1.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.1.2.3.4. タグ

- cluster.open-cluster-management.io

11.1.2.4. クラスターの削除

```
DELETE /cluster.open-cluster-management.io/v1/managedclusters/{cluster_name}
```

11.1.2.4.1. 説明

単一クラスターを削除します。

11.1.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**cluster_name** **必須**	削除するクラスターの名前。	文字列

11.1.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.1.2.4.4. タグ

- cluster.open-cluster-management.io

11.1.3. 定義

11.1.3.1. クラスター

名前	スキーマ
apiVersion **必須**	string
kind **必須**	string
metadata **必須**	object

名前	スキーマ
spec **必須**	spec

spec

名前	スキーマ
hubAcceptsClient **必須**	bool
managedClusterClientConfigs **任意**	< managedClusterClientConfigs > array
leaseDurationSeconds **任意**	整数 (int32)

managedClusterClientConfigs

名前	説明	スキーマ
URL **必須**		string
CABundle **任意**	**パターン**: **"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"**	文字列 (バイト)

11.2. CLUSTERSETS API (V1ALPHA1)

11.2.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンの Clusterset リソースを対象としています。Clusterset リソースには、create、query、delete、update の 4 つの要求を使用できます。

11.2.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.2.1.2. タグ

- cluster.open-cluster-management.io: Clustersets を作成して管理します。

11.2.2. パス

11.2.2.1. 全 clusterset のクエリー

```
GET /cluster.open-cluster-management.io/v1beta1/managedclustersets
```

11.2.2.1.1. 説明

Clustersets に対してクエリーを実行して詳細を確認します。

11.2.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.2.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.2.2.1.4. 消費

- **clusterset/yaml**

11.2.2.1.5. タグ

- cluster.open-cluster-management.io

11.2.2.2. clusterset の作成

```
POST /cluster.open-cluster-management.io/v1beta1/managedclustersets
```

11.2.2.2.1. 説明

Clusterset を作成します。

11.2.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body 必須**	作成する clusterset を記述するパラメーター	Clusterset

11.2.2.2.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.2.2.2.4. 消費

- **clusterset/yaml**

11.2.2.2.5. タグ

- cluster.open-cluster-management.io

11.2.2.2.6. HTTP 要求の例

11.2.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "ManagedClusterSet",
  "metadata" : {
    "name" : "clusterset1"
  },
  "spec": { },
  "status" : { }
}
```

11.2.2.3. 単一 clusterset のクエリー

GET /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}

11.2.2.3.1. 説明

単一の clusterset に対してクエリーを実行して詳細を確認します。

11.2.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterset_name** **必須**	問い合わせる clusterset の名前。	文字列

11.2.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.2.2.3.4. タグ

- cluster.open-cluster-management.io

11.2.2.4. clusterset の削除

DELETE /cluster.open-cluster-management.io/v1beta1/managedclustersets/{clusterset_name}

11.2.2.4.1. 説明

単一 clusterset を削除します。

11.2.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterset_name** **必須**	削除する clusterset の名前。	文字列

11.2.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.2.2.4.4. タグ

- cluster.open-cluster-management.io

11.2.3. 定義

11.2.3.1. Clusterset

名前	スキーマ
apiVersion **必須**	string
kind **必須**	string
metadata **必須**	object

11.3. CLUSTERVIEW API (V1ALPHA1)

11.3.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンの **clusterview** リソースを対象としています。**clusterview** リソースには、アクセス可能なマネージドクラスターおよびマネージドクラスターセットの一覧を表示できる CLI コマンドが含まれます。使用できる要求は、list、get、および watch の 3 つです。

11.3.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.3.1.2. タグ

- clusterview.open-cluster-management.io: お使いの ID がアクセスできるマネージドクラスターの一覧を表示します。

11.3.2. パス

11.3.2.1. マネージドクラスターの取得

```
GET /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.1.1. 説明

アクセス可能なマネージドクラスターの一覧を表示します。

11.3.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.3.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

HTTP コード	説明	スキーマ

11.3.2.1.4. 消費

- **managedcluster/yaml**

11.3.2.1.5. タグ

- clusterview.open-cluster-management.io

11.3.2.2. マネージドクラスターの一覧表示

```
LIST /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.2.1. 説明

アクセス可能なマネージドクラスターの一覧を表示します。

11.3.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body 任意**	マネージドクラスターを一覧表示するユーザー ID の名前	文字列

11.3.2.2.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.3.2.2.4. 消費

- **managedcluster/yaml**

11.3.2.2.5. タグ

- clusterview.open-cluster-management.io

11.3.2.2.6. HTTP 要求の例

11.3.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "clusterview.open-cluster-management.io/v1alpha1",
  "kind" : "ClusterView",
  "metadata" : {
    "name" : "<user_ID>"
  },
  "spec": { },
  "status" : { }
}
```

11.3.2.3. マネージドクラスターセットの監視

```
WATCH /managedclusters.clusterview.open-cluster-management.io
```

11.3.2.3.1. 説明

アクセス可能なマネージドクラスターを確認します。

11.3.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	文字列

11.3.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし

HTTP コード	説明	スキーマ
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.3.2.4. マネージドクラスターセットの一覧表示

```
GET /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.4.1. 説明

アクセス可能なマネージドクラスターを一覧表示します。

11.3.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	文字列

11.3.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.3.2.5. マネージドクラスターセットの一覧表示

```
LIST /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.5.1. 説明

アクセス可能なマネージドクラスターを一覧表示します。

11.3.2.5.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	文字列

11.3.2.5.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.3.2.6. マネージドクラスターセットの監視

```
WATCH /managedclustersets.clusterview.open-cluster-management.io
```

11.3.2.6.1. 説明

アクセス可能なマネージドクラスターを確認します。

11.3.2.6.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	文字列

11.3.2.6.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.4. CLUSTERSETBINDINGS API (V1ALPHA1)

11.4.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンの clustersetbinding リソースを対象としています。clustersetbinding リソースには、create、query、delete、update の 4 つの要求を使用できます。

11.4.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.4.1.2. タグ

- cluster.open-cluster-management.io: clustersetbinding を作成して管理します。

11.4.2. パス

11.4.2.1. 全 clustersetbinding のクエリー

```
GET /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings
```

-

11.4.2.1.1. 説明

clustersetbinding に対してクエリーを実行して詳細を確認します。

11.4.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**namespace** **必須**	使用する namespace (例: default)	文字列

11.4.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.4.2.1.4. 消費

- **clustersetbinding/yaml**

11.4.2.1.5. タグ

- cluster.open-cluster-management.io

11.4.2.2. clustersetbinding の作成

```
POST /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings
```

11.4.2.2.1. 説明

clustersetbinding を作成します。

11.4.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**namespace** **必須**	使用する namespace (例: default)	文字列
ボディー	**body** **必須**	作成する clustersetbinding を記述するパラメーター	Clustersetbinding

11.4.2.2.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.4.2.2.4. 消費

- **clustersetbinding/yaml**

11.4.2.2.5. タグ

- cluster.open-cluster-management.io

11.4.2.2.6. HTTP 要求の例

11.4.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1",
  "kind" : "ManagedClusterSetBinding",
  "metadata" : {
    "name" : "clusterset1",
    "namespace" : "ns1"
  },
 "spec": {
```

```
    "clusterSet": "clusterset1"
  },
  "status" : { }
}
```

11.4.2.3. 単一 clustersetbinding のクエリー

```
GET /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/managedclustersetbindings/{clustersetbinding_name}
```

11.4.2.3.1. 説明

単一の clustersetbinding に対してクエリーを実行して詳細を確認します。

11.4.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**namespace** **必須**	使用する namespace (例: default)	文字列
パス	**clustersetbinding_name** **必須**	問い合わせる clustersetbinding の名前	文字列

11.4.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.4.2.3.4. タグ

- cluster.open-cluster-management.io

11.4.2.4. clustersetbinding の削除

```
DELETE /cluster.open-cluster-
management.io/v1beta1/managedclustersetbindings/{clustersetbinding_name}
```

11.4.2.4.1. 説明

単一 clustersetbinding を削除します。

11.4.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**namespace** **必須**	使用する namespace (例: default)	文字列
パス	**clustersetbinding_name** **必須**	削除する clustersetbinding の名前	文字列

11.4.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.4.2.4.4. タグ

- cluster.open-cluster-management.io

11.4.3. 定義

11.4.3.1. Clustersetbinding

名前	スキーマ
apiVersion **必須**	string
kind **必須**	string
metadata **必須**	object
spec **必須**	spec

spec

名前	スキーマ
clusterSet **必須**	string

11.5. API

11.5.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジン用の MultiClusterEngine リソースを対象としています。**MultiClusterEngine** リソースには、create、query、delete、update の 4 つのリクエストがあります。

11.5.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.5.1.2. タグ

- multiclusterengines.multicluster.openshift.io : MultiClusterEngine を作成および管理します。

11.5.2. パス

11.5.2.1. MultiClusterEngine を作成する

```
POST /apis/multicluster.openshift.io/v1alpha1/multiclusterengines
```

11.5.2.1.1. 説明

MultiClusterEngine を作成します。

11.5.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body 必須**	作成する MultiClusterEngine を説明するパラメーター。	MultiClusterEngine

11.5.2.1.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.5.2.1.4. 消費

- **MultiClusterEngines/yaml**

11.5.2.1.5. タグ

- multiclusterengines.multicluster.openshift.io

11.5.2.1.5.1. 要求のボディー

```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "multiclusterengines.multicluster.openshift.io"
  },
  "spec": {
    "group": "multicluster.openshift.io",
    "names": {
      "kind": "MultiClusterEngine",
```

```
    "listKind": "MultiClusterEngineList",
    "plural": "multiclusterengines",
    "shortNames": [
      "mce"
    ],
    "singular": "multiclusterengine"
  },
  "scope": "Cluster",
  "versions": [
    {
      "additionalPrinterColumns": [
        {
          "description": "The overall state of the MultiClusterEngine",
          "jsonPath": ".status.phase",
          "name": "Status",
          "type": "string"
        },
        {
          "jsonPath": ".metadata.creationTimestamp",
          "name": "Age",
          "type": "date"
        }
      ],
      "name": "v1alpha1",
      "schema": {
        "openAPIV3Schema": {
          "description": "MultiClusterEngine is the Schema for the multiclusterengines\nAPI",
          "properties": {
            "apiVersion": {
              "description": "APIVersion defines the versioned schema of this representation\nof an
object. Servers should convert recognized schemas to the latest\ninternal value, and may reject
unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources",
              "type": "string"
            },
            "kind": {
              "description": "Kind is a string value representing the REST resource this\nobject
represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be
updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-
architecture/api-conventions.md#types-kinds",
              "type": "string"
            },
            "metadata": {
              "type": "object"
            },
            "spec": {
              "description": "MultiClusterEngineSpec defines the desired state of MultiClusterEngine",
              "properties": {
                "imagePullSecret": {
                  "description": "Override pull secret for accessing MultiClusterEngine\noperand and
endpoint images",
                  "type": "string"
                },
                "nodeSelector": {
                  "additionalProperties": {
                    "type": "string"
```

```
                },
                "description": "Set the nodeselectors",
                "type": "object"
              },
              "targetNamespace": {
                "description": "Location where MCE resources will be placed",
                "type": "string"
              },
              "tolerations": {
                "description": "Tolerations causes all components to tolerate any taints.",
                "items": {
                  "description": "The pod this Toleration is attached to tolerates any\ntaint that matches
the triple <key,value,effect> using the matching\noperator <operator>.",
                  "properties": {
                    "effect": {
                      "description": "Effect indicates the taint effect to match. Empty\nmeans match all taint
effects. When specified, allowed values\nare NoSchedule, PreferNoSchedule and NoExecute.",
                      "type": "string"
                    },
                    "key": {
                      "description": "Key is the taint key that the toleration applies\nto. Empty means match
all taint keys. If the key is empty,\noperator must be Exists; this combination means to match
all\nvalues and all keys.",
                      "type": "string"
                    },
                    "operator": {
                      "description": "Operator represents a key's relationship to the\nvalue. Valid operators
are Exists and Equal. Defaults to Equal.\nExists is equivalent to wildcard for value, so that a pod\ncan
tolerate all taints of a particular category.",
                      "type": "string"
                    },
                    "tolerationSeconds": {
                      "description": "TolerationSeconds represents the period of time\nthe toleration (which
must be of effect NoExecute, otherwise\nthis field is ignored) tolerates the taint. By default, it\nis not
set, which means tolerate the taint forever (do not\nevict). Zero and negative values will be treated as
0 (evict\nimmediately) by the system.",
                      "format": "int64",
                      "type": "integer"
                    },
                    "value": {
                      "description": "Value is the taint value the toleration matches\nto. If the operator is
Exists, the value should be empty,\notherwise just a regular string.",
                      "type": "string"
                    }
                  },
                  "type": "object"
                },
                "type": "array"
              }
            },
            "type": "object"
          },
          "status": {
            "description": "MultiClusterEngineStatus defines the observed state of MultiClusterEngine",
            "properties": {
              "components": {
```

```
                "items": {
                  "description": "ComponentCondition contains condition information for\ntracked
components",
                  "properties": {
                    "kind": {
                      "description": "The resource kind this condition represents",
                      "type": "string"
                    },
                    "lastTransitionTime": {
                      "description": "LastTransitionTime is the last time the condition\nchanged from one
status to another.",
                      "format": "date-time",
                      "type": "string"
                    },
                    "message": {
                      "description": "Message is a human-readable message indicating\ndetails about the
last status change.",
                      "type": "string"
                    },
                    "name": {
                      "description": "The component name",
                      "type": "string"
                    },
                    "reason": {
                      "description": "Reason is a (brief) reason for the condition's\nlast status change.",
                      "type": "string"
                    },
                    "status": {
                      "description": "Status is the status of the condition. One of True,\nFalse, Unknown.",
                      "type": "string"
                    },
                    "type": {
                      "description": "Type is the type of the cluster condition.",
                      "type": "string"
                    }
                  },
                  "type": "object"
                },
                "type": "array"
              },
              "conditions": {
                "items": {
                  "properties": {
                    "lastTransitionTime": {
                      "description": "LastTransitionTime is the last time the condition\nchanged from one
status to another.",
                      "format": "date-time",
                      "type": "string"
                    },
                    "lastUpdateTime": {
                      "description": "The last time this condition was updated.",
                      "format": "date-time",
                      "type": "string"
                    },
                    "message": {
                      "description": "Message is a human-readable message indicating\ndetails about the
```

```
last status change.",
                        "type": "string"
                      },
                      "reason": {
                        "description": "Reason is a (brief) reason for the condition's\nlast status change.",
                        "type": "string"
                      },
                      "status": {
                        "description": "Status is the status of the condition. One of True,\nFalse, Unknown.",
                        "type": "string"
                      },
                      "type": {
                        "description": "Type is the type of the cluster condition.",
                        "type": "string"
                      }
                    },
                    "type": "object"
                  },
                  "type": "array"
                },
                "phase": {
                  "description": "Latest observed overall state",
                  "type": "string"
                }
              },
              "type": "object"
            }
          },
          "type": "object"
        }
      },
      "served": true,
      "storage": true,
      "subresources": {
        "status": {}
      }
    }
  ]
 },
 "status": {
  "acceptedNames": {
    "kind": "",
    "plural": ""
  },
  "conditions": [],
  "storedVersions": []
 }
}
```


11.5.2.2. すべての MultiClusterEngine をクエリーする

```
GET /apis/multicluster.openshift.io/v1alpha1/multiclusterengines
```

11.5.2.2.1. 説明

詳細については、マルチクラスターエンジンに問い合わせてください。

11.5.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.5.2.2.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.5.2.2.4. 消費

- **operator/yaml**

11.5.2.2.5. タグ

- multiclusterengines.multicluster.openshift.io

11.5.2.3. MultiClusterEngine Operator の削除

```
DELETE /apis/multicluster.openshift.io/v1alpha1/multiclusterengines/{name}
```

11.5.2.3.1. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**name** **必須**	削除するマルチクラスターエンジンの名前。	文字列

11.5.2.3.2. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.5.2.3.3. タグ

- multiclusterengines.multicluster.openshift.io

11.5.3. 定義

11.5.3.1. MultiClusterEngine

名前	説明	スキーマ
apiVersion **必須**	MultiClusterEngines のバージョン管理されたスキーマ。	string
kind **必須**	REST リソースを表す文字列の値	string
metadata **必須**	リソースを定義するルールを記述します。	object
spec **必須**	MultiClusterEngineSpec は、MultiClusterEngine の望ましい状態を定義します。	**仕様の一覧** を参照してください。

11.5.3.2. 仕様の一覧

名前	説明	スキーマ
nodeSelector **任意**	nodeselectors を設定します。	map[string]string

名前	説明	スキーマ
imagePullSecret **任意**	MultiClusterEngine オペランドおよびエンドポイントイメージにアクセスするためのプルシークレットをオーバーライドします。	string
tolerations **任意**	許容範囲により、すべてのコンポーネントがあらゆる Taint を許容します。	[]corev1.Toleration
targetNamespace **optional**	MCE リソースが配置される場所。	文字列

11.6. PLACEMENTS API (V1ALPHA1)

11.6.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンの配置リソースに関するものです。Placement リソースには、create、query、delete、update の 4 つの要求を使用できます。

11.6.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.6.1.2. タグ

- cluster.open-cluster-management.io: Placement を作成して管理します。

11.6.2. パス

11.6.2.1. 全 Placement のクエリー

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements
```

11.6.2.1.1. 説明

Placement に対してクエリーを実行して詳細を確認します。

11.6.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.6.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.6.2.1.4. 消費

- **placement/yaml**

11.6.2.1.5. タグ

- cluster.open-cluster-management.io

11.6.2.2. Placement の作成

```
POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placements
```

11.6.2.2.1. 説明

Placement を作成します。

11.6.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body 必須**	作成する placement を記述するパラメーター	Placement

11.6.2.2.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.6.2.2.4. 消費

- **placement/yaml**

11.6.2.2.5. タグ

- cluster.open-cluster-management.io

11.6.2.2.6. HTTP 要求の例

11.6.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "Placement",
  "metadata" : {
    "name" : "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status" : { }
}
```

11.6.2.3. 単一の Placement のクエリー

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}
```

11.6.2.3.1. 説明

1 つの Placement に対してクエリーを実行して詳細を確認します。

11.6.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**placement_name** **必須**	問い合わせる Placement の名前	文字列

11.6.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.6.2.3.4. タグ

- cluster.open-cluster-management.io

11.6.2.4. Placement の削除

```
DELETE /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placements/{placement_name}
```

11.6.2.4.1. 説明

単一の Placement を削除します。

11.6.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**placement_name** **必須**	削除する Placement の名前	文字列

11.6.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.6.2.4.4. タグ

- cluster.open-cluster-management.io

11.6.3. 定義

11.6.3.1. Placement

名前	説明	スキーマ
apiVersion **必須**	Placement のバージョンスキーマ	string
kind **必須**	REST リソースを表す文字列の値	string
metadata **必須**	Placement のメタデータ	object
spec **必須**	Placement の仕様	spec

spec

名前	説明	スキーマ
ClusterSets **任意**	ManagedClusters を選択する ManagedClusterSets のサブセット。空白の場合には、Placement namespace にバインドされる ManagedClusterSets から ManagedClusters が選択されます。それ以外の場合は、ManagedClusters がこのサブセットの交差部分から選択され、ManagedClusterSets は Placement namespace にバインドされます。	string array
numberOfClusters **任意**	選択する ManagedClusters の必要数	整数 (int32)
predicates **任意**	ManagedClusters を選択するクラスター述語のサブセット。条件ロジックは **OR** です。	clusterPredicate アレイ

clusterPredicate

名前	説明	スキーマ
requiredClusterSelector **任意**	ラベルおよびクラスター要求のある ManagedClusters を選択するクラスターセレクター	clusterSelector

clusterSelector

名前	説明	スキーマ
labelSelector **任意**	ラベル別の ManagedClusters のセレクター	object
claimSelector **任意**	要求別の ManagedClusters のセレクター	clusterClaimSelector

clusterClaimSelector

名前	説明	スキーマ

名前	説明	スキーマ
matchExpressions **任意**	クラスター要求のセレクター要件のサブセット。条件ロジックは **AND** です。	< オブジェクト > 配列

11.7. PLACEMENTDECISIONS API (V1ALPHA1)

11.7.1. 概要

このドキュメントは、Kubernetes のマルチクラスターエンジンの PlacementDecision リソースを対象としています。PlacementDecision リソースには、create、query、delete、update の 4 つの要求を使用できます。

11.7.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.7.1.2. タグ

- cluster.open-cluster-management.io: PlacementDecision を作成して管理します。

11.7.2. パス

11.7.2.1. 全 PlacementDecision のクエリー

```
GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions
```

11.7.2.1.1. 説明

PlacementDecisions に対してクエリーを実行して詳細を確認します。

11.7.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

11.7.2.1.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし

HTTP コード	説明	スキーマ
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.7.2.1.4. 消費

- **placementdecision/yaml**

11.7.2.1.5. タグ

- cluster.open-cluster-management.io

11.7.2.2. PlacementDecision の作成

```
POST /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions
```

11.7.2.2.1. 説明

PlacementDecisions を作成します。

11.7.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body 必須**	作成する PlacementDecision を記述するパラメーター	PlacementDecision

11.7.2.2.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし

HTTP コード	説明	スキーマ
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.7.2.2.4. 消費

- **placementdecision/yaml**

11.7.2.2.5. タグ

- cluster.open-cluster-management.io

11.7.2.2.6. HTTP 要求の例

11.7.2.2.6.1. 要求のボディー

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1alpha1",
  "kind" : "PlacementDecision",
  "metadata" : {
    "labels" : {
      "cluster.open-cluster-management.io/placement" : "placement1"
    },
    "name" : "placement1-decision1",
    "namespace": "ns1"
  },
  "status" : { }
}
```

11.7.2.3. 単一の PlacementDecision のクエリー

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.3.1. 説明

1 つの PlacementDecisions に対してクエリーを実行して詳細を確認します。

11.7.2.3.2. パラメーター

型	名前	説明	スキーマ

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**placementdecision_name 必須**	問い合わせる PlacementDecision の名前	文字列

11.7.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.7.2.3.4. タグ

- cluster.open-cluster-management.io

11.7.2.4. PlacementDecision の削除

```
DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

11.7.2.4.1. 説明

単一の PlacementDecision を削除します。

11.7.2.4.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列

型	名前	説明	スキーマ
パス	**placementdecision_name** **必須**	削除する PlacementDecision の名前	文字列

11.7.2.4.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.7.2.4.4. タグ

- cluster.open-cluster-management.io

11.7.3. 定義

11.7.3.1. PlacementDecision

名前	説明	スキーマ
apiVersion **必須**	PlacementDecision のバージョンスキーマ	string
kind **必須**	REST リソースを表す文字列の値	string
metadata **必須**	PlacementDecision のメタデータ	オブジェクト

11.8. マネージドサービスアカウント (テクノロジープレビュー)

11.8.1. 概要

このドキュメントは、Kubernetes Operator のマルチクラスターエンジンの **ManagedServiceAccount** リソースを対象としています。**ManagedServiceAccount** リソースには、create、query、delete、update の 4 つのリクエストがあります。

11.8.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

11.8.1.2. タグ

- **managedserviceaccounts.multicluster.openshift.io`**: **ManagedServiceAccounts** を作成および管理します

11.8.2. パス

11.8.2.1. ManagedServiceAccount を作成する

```
POST /apis/multicluster.openshift.io/v1alpha1/ManagedServiceAccounts
```

11.8.2.1.1. 説明

ManagedServiceAccount を作成します。

11.8.2.1.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
ボディー	**body** **必須**	作成する ManagedServiceAccount を説明するパラメーター。	ManagedServiceAccount

11.8.2.1.3. 応答

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし

HTTP コード	説明	スキーマ
503	サービスが利用できない	コンテンツなし

11.8.2.1.4. 消費

- **managedserviceaccount/yaml**

11.8.2.1.5. タグ

- managedserviceaccount.multicluster.openshift.io

11.8.2.1.5.1. 要求のボディー


```
{
  "apiVersion": "apiextensions.k8s.io/v1",
  "kind": "CustomResourceDefinition",
  "metadata": {
    "annotations": {
      "controller-gen.kubebuilder.io/version": "v0.4.1"
    },
    "creationTimestamp": null,
    "name": "managedserviceaccount.authentication.open-cluster-management.io"
  },
  "spec": {
    "group": "authentication.open-cluster-management.io",
    "names": {
      "kind": "ManagedServiceAccount",
      "listKind": "ManagedServiceAccountList",
      "plural": "managedserviceaccounts",
      "singular": "managedserviceaccount"
    },
    "scope": "Namespaced",
    "versions": [
      {
        "name": "v1alpha1",
        "schema": {
          "openAPIV3Schema": {
            "description": "ManagedServiceAccount is the Schema for the
managedserviceaccounts\nAPI",
            "properties": {
              "apiVersion": {
                "description": "APIVersion defines the versioned schema of this representation\nof an
object. Servers should convert recognized schemas to the latest\ninternal value, and may reject
unrecognized values. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-
conventions.md#resources",
                "type": "string"
              },
              "kind": {
                "description": "Kind is a string value representing the REST resource this\nobject
represents. Servers may infer this from the endpoint the client\nsubmits requests to. Cannot be
updated. In CamelCase. More info: https://git.k8s.io/community/contributors/devel/sig-
```

```
architecture/api-conventions.md#types-kinds",
            "type": "string"
          },
          "metadata": {
            "type": "object"
          },
          "spec": {
            "description": "ManagedServiceAccountSpec defines the desired state of
ManagedServiceAccount",
            "properties": {
              "rotation": {
                "description": "Rotation is the policy for rotation the credentials.",
                "properties": {
                  "enabled": {
                    "default": true,
                    "description": "Enabled prescribes whether the ServiceAccount token\nwill be rotated
from the upstream",
                    "type": "boolean"
                  },
                  "validity": {
                    "default": "8640h0m0s",
                    "description": "Validity is the duration for which the signed ServiceAccount\ntoken is
valid.",
                    "type": "string"
                  }
                },
                "type": "object"
              },
              "ttlSecondsAfterCreation": {
                "description": "ttlSecondsAfterCreation limits the lifetime of a
ManagedServiceAccount.\nIf the ttlSecondsAfterCreation field is set, the
ManagedServiceAccount\nwill be automatically deleted regardless of the
ManagedServiceAccount's\nstatus. When the ManagedServiceAccount is deleted, its
lifecycle\nguarantees (e.g. finalizers) will be honored. If this field is unset,\nthe
ManagedServiceAccount won't be automatically deleted. If this\nfield is set to zero, the
ManagedServiceAccount becomes eligible\nfor deletion immediately after its creation. In order to use
ttlSecondsAfterCreation,\nthe EphemeralIdentity feature gate must be enabled.",
                "exclusiveMinimum": true,
                "format": "int32",
                "minimum": 0,
                "type": "integer"
              }
            },
            "required": [
              "rotation"
            ],
            "type": "object"
          },
          "status": {
            "description": "ManagedServiceAccountStatus defines the observed state
of\nManagedServiceAccount",
            "properties": {
              "conditions": {
                "description": "Conditions is the condition list.",
                "items": {
                  "description": "Condition contains details for one aspect of the current\nstate of this API
```

Resource. --- This struct is intended for direct\nuse as an array at the field path .status.conditions. For example,\ntype FooStatus struct{ // Represents the observations of a\nfoo's current state. // Known .status.conditions.type are:\n\"Available\", \"Progressing\", and \"Degraded\" // +patchMergeKey=type\n // +patchStrategy=merge // +listType=map // +listMapKey=type\n Conditions []metav1.Condition `json:\"conditions,omitempty\"\npatchStrategy:\"merge\" patchMergeKey:\"type\" protobuf:\"bytes,1,rep,name=conditions\"`\n\n // other fields }",
"properties": {
"lastTransitionTime": {
"description": "lastTransitionTime is the last time the condition\ntransitioned from one status to another. This should be when\nthe underlying condition changed. If that is not known, then\nusing the time when the API field changed is acceptable.",
"format": "date-time",
"type": "string"
},
"message": {
"description": "message is a human readable message indicating\ndetails about the transition. This may be an empty string.",
"maxLength": 32768,
"type": "string"
},
"observedGeneration": {
"description": "observedGeneration represents the .metadata.generation\nthat the condition was set based upon. For instance, if .metadata.generation\nis currently 12, but the .status.conditions[x].observedGeneration\nis 9, the condition is out of date with respect to the current\nstate of the instance.",
"format": "int64",
"minimum": 0,
"type": "integer"
},
"reason": {
"description": "reason contains a programmatic identifier indicating\nthe reason for the condition's last transition. Producers\nof specific condition types may define expected values and\nmeanings for this field, and whether the values are considered\na guaranteed API. The value should be a CamelCase string.\nThis field may not be empty.",
"maxLength": 1024,
"minLength": 1,
"pattern": "^[A-Za-z]([A-Za-z0-9_,:]*[A-Za-z0-9_])?$",
"type": "string"
},
"status": {
"description": "status of the condition, one of True, False, Unknown.",
"enum": [
"True",
"False",
"Unknown"
],
"type": "string"
},
"type": {
"description": "type of condition in CamelCase or in foo.example.com/CamelCase.\n--- Many .condition.type values are consistent across resources\nlike Available, but because arbitrary conditions can be useful\n(see .node.status.conditions), the ability to deconflict is\nimportant. The regex it matches is (dns1123SubdomainFmt/)?(qualifiedNameFmt)",
"maxLength": 316,
"pattern": "^([a-z0-9]([-a-z0-9]*[a-z0-9])?(\\.[a-z0-9]([-a-z0-9]*[a-z0-9])?)*/)?(([A-Za-z0-9][-A-Za-z0-9_.]*)?[A-Za-z0-9])$",

```
                    "type": "string"
                   }
                  },
                  "required": [
                   "lastTransitionTime",
                   "message",
                   "reason",
                   "status",
                   "type"
                  ],
                  "type": "object"
                 },
                 "type": "array"
                },
                "expirationTimestamp": {
                 "description": "ExpirationTimestamp is the time when the token will expire.",
                 "format": "date-time",
                 "type": "string"
                },
                "tokenSecretRef": {
                 "description": "TokenSecretRef is a reference to the corresponding
ServiceAccount's\nSecret, which stores the CA certficate and token from the managed\ncluster.",
                 "properties": {
                  "lastRefreshTimestamp": {
                   "description": "LastRefreshTimestamp is the timestamp indicating\nwhen the token in
the Secret is refreshed.",
                   "format": "date-time",
                   "type": "string"
                  },
                  "name": {
                   "description": "Name is the name of the referenced secret.",
                   "type": "string"
                  }
                 },
                 "required": [
                  "lastRefreshTimestamp",
                  "name"
                 ],
                 "type": "object"
                }
               },
               "type": "object"
              }
             },
             "type": "object"
            }
           },
           "served": true,
           "storage": true,
           "subresources": {
            "status": {}
           }
          }
         ]
        },
        "status": {
```

```
    "acceptedNames": {
      "kind": "",
      "plural": ""
    },
    "conditions": [],
    "storedVersions": []
  }
}
```

11.8.2.2. 単一の ManagedServiceAccount をクエリーする

```
GET /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccoun
t_name}
```

11.8.2.2.1. 説明

詳細については、単一の **ManagedServiceAccount** を照会してください。

11.8.2.2.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE** **必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**managedserviceaccount_name** **required**	照会する **ManagedServiceAccount** の名前。	文字列

11.8.2.2.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.8.2.2.4. タグ

- cluster.open-cluster-management.io

11.8.2.3. ManagedServiceAccount を削除する

```
DELETE /cluster.open-cluster-
management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccoun
t_name}
```

11.8.2.3.1. 説明

単一の **ManagedServiceAccount** を削除します。

11.8.2.3.2. パラメーター

型	名前	説明	スキーマ
ヘッダー	**COOKIE 必須**	Authorization: Bearer {ACCESS_TOKEN}。ACCESS_TOKEN はユーザーのアクセストークンに置き換えます。	文字列
パス	**managedserviceaccount_name required**	削除する **ManagedServiceAccount** の名前。	文字列

11.8.2.3.3. レスポンス

HTTP コード	説明	スキーマ
200	成功	コンテンツなし
403	アクセス禁止	コンテンツなし
404	リソースが見つからない	コンテンツなし
500	内部サービスエラー	コンテンツなし
503	サービスが利用できない	コンテンツなし

11.8.2.3.4. タグ

- cluster.open-cluster-management.io

11.8.3. 定義

11.8.3.1. ManagedServiceAccount

名前	説明	スキーマ
apiVersion **必須**	**ManagedServiceAccount** のバージョン管理されたスキーマ。	string
kind **必須**	REST リソースを表す文字列の値	string
metadata **必須**	**ManagedServiceAccount** のメタデータ。	object
spec **必須**	**ManagedServiceAccount** の仕様。	

第12章 アンインストール

Kubernetes のマルチクラスターエンジンをアンインストールすると、プロセスの 2 つの異なるレベルが表示されます。**custom resource removal** と **complete operator uninstall** です。アンインストールプロセスの完了に最長 5 分かかる可能性があります。

- カスタムリソースの削除は、最も基本的なアンインストールの種類で、**MultiClusterEngine** インスタンスのカスタムリソースを削除しますが、他の必要なコンポーネントが残されたままになります。このレベルのアンインストールは、同じ設定とコンポーネントを使用して再インストールする予定の場合に役立ちます。

- 2 番目のレベルは、より完全なアンインストールで、カスタムリソース定義などのコンポーネントを除き、ほとんどの Operator コンポーネントを削除します。この手順を続行すると、カスタムリソースの削除で削除されていないコンポーネントおよびサブスクリプションがすべて削除されます。アンインストールが済むと、カスタムリソースの前に Operator を再インストールする必要があります。

12.1. 前提条件: 有効化されたサービスのデタッチ

Kubernetes エンジンのマルチクラスターエンジンをアンインストールする前に、そのエンジンによって管理されているすべてのクラスターをデタッチする必要があります。エラーを回避するには、エンジンによって管理されているすべてのクラスターをデタッチしてから、アンインストールを再試行してください。

- マネージドクラスターがアタッチされている場合は、以下のメッセージが表示される可能性があります。

  ```
  Cannot delete MultiClusterEngine resource because ManagedCluster resource(s) exist
  ```

  クラスターのデタッチの詳細は、クラスターの作成 でお使いのプロバイダーの情報を選択して、**マネージメントからのクラスターの削除** セクションを参照してください。

12.2. コマンドを使用したリソースの削除

1. まだの場合には、**oc** コマンドが実行できるように、OpenShift Container Platform CLI が設定されていることを確認してください。**oc** コマンドの設定方法は、Red Hat OpenShift Container Platform ドキュメントの OpenShift CLI の使用方法 を参照してください。

2. 以下のコマンドを入力してプロジェクトの namespace に移動します。**namespace** はお使いのプロジェクトの namespace 名に置き換えます。

   ```
   oc project <namespace>
   ```

3. 次のコマンドを入力して、**MultiClusterEngine** カスタムリソースを削除します。

   ```
   oc delete multiclusterengine --all
   ```

   以下のコマンドを入力して進捗を表示できます。

   ```
   oc get multiclusterengine -o yaml
   ```

4. 以下のコマンドを入力し、インストールされている namespace の multicluster-engine **ClusterServiceVersion** を削除します。

```
❯ oc get csv
NAME                         DISPLAY                           VERSION   REPLACES   PHASE
multicluster-engine.v2.0.0   multicluster engine for Kubernetes   2.0.0               Succeeded

❯ oc delete clusterserviceversion multicluster-engine.v2.0.0
❯ oc delete sub multicluster-engine
```

ここに表示されている CSV バージョンは異なる場合があります。

12.3. コンソールを使用したコンポーネントの削除

Red Hat OpenShift Container Platform コンソールを使用してアンインストールする場合に、operator を削除します。コンソールを使用してアンインストールを行うには、以下の手順を実行します。

1. OpenShift Container Platform コンソールナビゲーションで、**Operators** > **Installed Operators** > **multicluster engine for Kubernetes**を選択します。
2. **MultiClusterEngine** カスタムリソースを削除します。
   a. **Multiclusterengine** のタブを選択します。
   b. MultiClusterEngine カスタムリソースの **Options** メニューを選択します。
   c. **Delete MultiClusterEngine** を選択します。
3. 次のセクションの手順に従って、クリーンアップスクリプトを実行します。
   ヒント: Kubernetes バージョンで同じマルチクラスターエンジンを再インストールする場合は、この手順の残りの手順をスキップして、カスタムリソースを再インストールできます。
4. **Installed Operators** に移動します。
5. **Options** メニューから **Uninstall operator** を選択して、_ multicluster engine for Kubernetes_ Operator を削除してください。

12.4. トラブルシューティングアンインストール

マルチクラスターエンジンのカスタムリソースが削除されていない場合は、クリーンアップスクリプトを実行して、残っている可能性のあるアーティファクトをすべて削除します。

a. 以下のスクリプトをファイルにコピーします。

```
#!/bin/bash
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete validatingwebhookconfiguration multiclusterengines.multicluster.openshift.io
oc delete mce --all
```

第13章 MUST-GATHER コマンドを実行したトラブルシューティング

トラブルシューティングを開始するには、問題のデバッグを行う **must-gather** コマンドを実行する場合のトラブルシューティングシナリオについて確認し、このコマンドの使用を開始する手順を参照してください。

必要なアクセス権限: クラスターの管理者

13.1. MUST-GATHER のシナリオ

- **シナリオ 1: 文書化されたトラブルシューティング** セクションを使用して、問題の解決策がまとめられているかどうかを確認します。本ガイドは、製品の主な機能別に設定されています。このシナリオでは、解決策が本書にまとめられているかどうかを、本ガイドで確認します。
- **シナリオ 2:** 問題の解決策の手順が文書にまとめられていない場合は、**must-gather** コマンドを実行し、その出力を使用して問題をデバッグします。
- **シナリオ 3: must-gather** コマンドの出力を使用して問題をデバッグできない場合は、出力を Red Hat サポートに共有します。

13.2. MUST-GATHER の手順

must-gather コマンドの使用を開始するには、以下の手順を参照してください。

1. **must-gather** コマンドについて確認し、Red Hat OpenShift Container Platform の クラスターに関するデータの収集 に必要な前提条件をインストールします。
2. クラスターにログインします。通常のユースケースでは、**engine** クラスターにログインして、**must-gather** を実行する必要があります。
   注記: マネージドクラスターを確認する場合は、**cluster-scoped-resources** ディレクトリーにある **gather-managed.log** ファイルを検索します。

   ```
   <your-directory>/cluster-scoped-resources/gather-managed.log>
   ```

   JOINED および AVAILABLE 列に **True** が設定されていないマネージドクラスターがないかを確認します。**must-gather** コマンドは、ステータスが **True** として関連付けられていないクラスター上で、実行できます。
3. データとディレクトリーの収集に使用される Kubernetes イメージのマルチクラスターエンジンを追加します。以下のコマンドを実行して、出力用にイメージとディレクトリーを挿入します。

   ```
   oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.0.0 --dest-dir=<directory>
   ```
4. 指定したディレクトリーに移動し、以下のレベルに整理されている出力を確認します。
   - ピアレベル 2 つ: **cluster-scoped-resources** と **namespace** のリソース
   - それぞれに対するサブレベル: クラススコープおよび namespace スコープの両方のリソースに対するカスタムリソース定義の API グループ。
   - それぞれに対する次のレベル: **kind** でソートされた YAML ファイル

13.3. 非接続環境での MUST-GATHER

非接続環境で **must-gather** コマンドを実行するには、次の手順を実行します。

1. 非接続環境では、Red Hat Operator のカタログイメージをミラーレジストリーにミラーリングします。詳細は、ネットワーク切断状態でのインストール を参照してください。
2. 次のコマンドを実行して、ミラーレジストリーからイメージを参照するログを抽出します。

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-rhel8@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8

oc adm must-gather --image=$IMAGE --dest-dir=./data
```

ここ で製品チームのバグを開くことができます。