Red Hat Advanced Cluster Management for Kubernetes 2.10

クラスター

クラスター管理

Last Updated: 2024-05-24

Red Hat Advanced Cluster Management for Kubernetes 2.10 クラスター

クラスター管理

法律上の通知

概要

クラスターライフサイクル (別称: Multi-Cluster Engine Operator) を使用すると、クラスターを作成および管理できます。このガイドでは、クラスター管理タスク、リリースノート、およびトラブルシューティング情報にアクセスできます。

目次

第1章 MULTICLUSTER ENGINE OPERATOR を使用したクラスターライフサイクルについて

multicluster engine Operator は、OpenShift Container Platform および Red Hat Advanced Cluster Management ハブクラスターにクラスター管理機能を提供するクラスターライフサイクル Operator です。ハブクラスターから、クラスターを作成および管理し、作成したクラスターを破棄できます。クラスターを休止、再開、およびデタッチすることもできます。クラスターライフサイクル機能の詳細は、以下のドキュメントを参照してください。

ハブクラスターとマネージドクラスターの要件とサポートについては、サポートマトリックス にアクセスしてください。

情報:

- クラスターは、Hive リソースとともに OpenShift Container Platform クラスターインストーラーを使用して作成されます。OpenShift Container Platform クラスターをインストールするプロセスについての詳細は、OpenShift Container Platform ドキュメントの OpenShift Container Platform インストールの概要 を参照してください。
- OpenShift Container Platform クラスターでは、multicluster engine Operator をクラスターライフサイクル機能のスタンドアロンクラスターマネージャーとして使用するか、Red Hat Advanced Cluster Management ハブクラスターの一部として使用できます。
- OpenShift Container Platform のみを使用している場合、Operator はサブスクリプションに含まれます。OpenShift Container Platform ドキュメントから、Kubernetes Operator 用のマルチクラスターエンジンについて を参照してください。
- Red Hat Advanced Cluster Management にサブスクライブすると、インストールとともに Operator も受信されます。Red Hat Advanced Cluster Management ハブクラスターを使用して、他の Kubernetes クラスターを作成、管理、および監視できます。Red Hat Advanced Cluster Management インストールおよびアップグレード に関するドキュメント を参照してください。
- リリースイメージは、クラスターの作成時に使用する OpenShift Container Platform のバージョンです。Red Hat Advanced Cluster Management を使用して作成されたクラスターの場合、リリースイメージの自動アップグレードを有効にできます。Red Hat Advanced Cluster Management のリリースイメージの詳細は、リリースイメージ を参照してください。
    - multicluster engine operator を使用したクラスターライフサイクルについて
    - リリースノート
    - multicluster engine operator のインストールとアップグレード
    - 認証情報の管理
    - クラスターライフサイクルの概要
    - Discovery サービスの概要
    - Hosted Control Plane
    - API
    - トラブルシューティング

クラスターライフサイクル管理アーキテクチャーのコンポーネントは、クラスターライフサイクルアーキテクチャー に含まれています。

1.1. リリースノート

現在のリリースについて学びます。

注記: Red Hat Advanced Cluster Management の 2.6 以前のバージョンはサービスから **削除** され、サポートされなくなりました。バージョン 2.6 以前のドキュメントは更新されていません。ドキュメントはそのまま利用できますが、エラータやその他の更新はなく、非推奨となります。

- multicluster engine operator の新機能
- エラータの更新
- クラスターライフサイクルの既知の問題
- 非推奨と削除

現在サポートされているリリースのいずれか、製品ドキュメントで問題が発生した場合は、Red Hat サポート にアクセスして、トラブルシューティングを行ったり、**ナレッジベース** の記事を表示したり、サポートチームに連絡したり、ケースを開いたりすることができます。認証情報でログインする必要があります。

Red Hat Customer PortalFAQ で、カスタマーポータルのドキュメントの詳細を確認することもできます。

1.1.1. multicluster engine operator を使用したクラスターライフサイクルの新機能

重要: 一部の機能およびコンポーネントは テクノロジープレビュー として指定され、リリースされます。

詳細は、本リリースの新機能を参照してください。

- クラスターライフサイクル
- Credentials
- Hosted Control Plane
- Red Hat Advanced Cluster Management の統合

1.1.1.1. クラスターライフサイクル

multicluster engine operator とクラスターライフサイクルに関連する新機能について説明します。

- マネージドクラスターが HTTP および HTTPS プロキシーサーバー経由でハブクラスターと通信できるように、クラスタープロキシーアドオンのプロキシー設定を指定できるようになりました。詳細は、クラスタープロキシーアドオンのプロキシー設定 を参照してください。
- **ManagedServiceAccount** アドオンがデフォルトで有効になるようになりました。multicluster engine Operator バージョン 2.4 からアップグレードする場合にアドオンを有効にする方法の詳細は、ManagedServiceAccount アドオンの有効化 を参照してください。
- **テクノロジープレビュー:** サーバー URL とハブクラスター API CA バンドルをカスタマイズできるようになりました。これにより、中間コンポーネントがある場合でも、multicluster engine

Operator ハブクラスターにマネージドクラスターをインポートできるようになります。詳細は、サーバー URL とハブクラスター API CA バンドルのカスタマイズ (テクノロジープレビュー) を参照してください。

- 各リクエストで HTTP/HTTPS ヘッダーとクエリーパラメーターを渡して、OS イメージを取得できるようになりました。詳細は、非接続環境での central infrastructure management の有効化 を参照してください。
- 認証に自己署名証明書またはサードパーティーの CA 証明書を使用して、TLS が有効な HTTPS **osImages** を保存およびダウンロードできるようになりました。詳細は、非接続環境での central infrastructure management の有効化 を参照してください。

1.1.1.2. Credentials

- 統合されたコンソールを使用して、**Cluster OS image** フィールドを設定し、VMware vSphere での非接続インストール用に認証情報を作成できるようになりました。詳細は、コンソールを使用した認証情報の管理 を参照してください。

1.1.1.3. Hosted Control Plane

- **テクノロジープレビュー:** 非ベアメタルエージェントマシンを使用して、Hosted Control Plane クラスターをプロビジョニングできます。詳細は、非ベアメタルエージェントマシンを使用した Hosted Control Plane クラスターの設定 を参照してください。
- コンソールを使用して、ホステッドクラスターを KubeVirt プラットフォームで作成できるようになりました。詳細は、コンソールを使用したホステッドクラスターの作成 を参照してください。
- 追加のネットワークの設定、仮想マシン (VM) 用の Guaranteed CPU へのアクセス要求、およびノードプールの KubeVirt 仮想マシンのスケジュール管理を実行できるようになりました。詳細は、追加のネットワーク、Guaranteed CPU、およびノードプールの仮想マシンのスケジュールを設定する を参照してください。

1.1.1.4. Red Hat Advanced Cluster Management の統合

Red Hat Advanced Cluster Management のインストール後に可観測性を有効にすると、Grafana ダッシュボードを使用して、Hosted Control Plane クラスター容量の推定値と既存の Hosted Control Plane のリソース使用率を表示できます。詳細は、Red Hat Advanced Cluster Management の統合 を参照してください。

1.1.2. クラスターライフサイクルの既知の問題

multicluster engine operator を使用したクラスターライフサイクルの既知の問題を確認します。以下のリストには、本リリースの既知の問題、または以前のリリースから持ち越された既知の問題が記載されています。OpenShift Container Platform クラスターについては、OpenShift Container Platform リリースノート を参照してください。

- クラスターライフサイクル
- Hosted Control Plane

1.1.2.1. クラスター管理

クラスターライフサイクルの既知の問題と制限は、multicluster engine Operator のドキュメントを使用したクラスターライフサイクルの一部です。

1.1.2.1.1. nmstate の制限事項

コピーアンドペースト機能を設定することで、開発を迅速化します。**assisted-installer** で **copy-from-mac** 機能を設定するには、**nmstate** 定義インターフェイスと **mac-mapping** インターフェイスに **mac-address** を追加する必要があります。**mac-mapping** インターフェイスは、**nmstate** 定義インターフェイスの外部で提供されます。そのため、同じ **mac-address** を 2 回指定する必要があります。

1.1.2.1.2. プリフックに問題があっても、ホステッドクラスターの作成は失敗しない

ホステッドクラスターの作成に自動化テンプレートを使用し、プリフックジョブが失敗した場合は、ホステッドクラスターの作成がまだ進行中であるように見えます。ホステッドクラスターは完全な障害状態を想定して設計されていないため、クラスターの作成を試行し続けるため、これは正常です。

1.1.2.1.3. アドオンの削除時にマネージドクラスターで必要な VolSync CSV の手動削除

ハブクラスターから VolSync **ManagedClusterAddOn** を削除すると、マネージドクラスターの VolSync Operator サブスクリプションが削除されますが、クラスターサービスバージョン (CSV) は削除されません。マネージドクラスターから CSV を削除するには、VolSync を削除する各マネージドクラスターで以下のコマンドを実行します。

```
oc delete csv -n openshift-operators volsync-product.v0.6.0
```

別のバージョンの VolSync がインストールされている場合は、**v0.6.0** をインストール済みバージョンに置き換えます。

1.1.2.1.4. マネージドクラスターセットを削除してもそのラベルが自動的に削除されない

ManagedClusterSet を削除した後に、クラスターセットに関連付ける各マネージドクラスターに追加されるラベルは自動的に削除されません。削除したマネージドクラスターセットに含まれる各マネージドクラスターからラベルを手動で削除します。ラベルは **cluster.open-cluster-management.io/clusterset:<ManagedClusterSet Name>** のようになります。

1.1.2.1.5. ClusterClaim エラー

ClusterPool に対して Hive **ClusterClaim** を作成し、**ClusterClaimspec** ライフタイムフィールドを無効な golang 時間値に手動で設定すると、製品は不正な要求だけでなく、すべての **ClusterClaims** を満たし、調整を停止します。

このエラーが発生すると、**clusterclaim-controller** Pod ログに以下の内容が表示されます。これは、プール名と、無効な有効期限が含まれた特定の例です。

```
E0203 07:10:38.266841       1 reflector.go:138] sigs.k8s.io/controller-runtime/pkg/cache/internal/informers_map.go:224: Failed to watch *v1.ClusterClaim: failed to list *v1.ClusterClaim: v1.ClusterClaimList.Items: []v1.ClusterClaim: v1.ClusterClaim.v1.ClusterClaim.Spec: v1.ClusterClaimSpec.Lifetime: unmarshalerDecoder: time: unknown unit "w" in duration "1w", error found in #10 byte of ...|time":"1w"}},{"apiVe|..., bigger context ...|clusterPoolName":"policy-aas-hubs","lifetime":"1w"}}, {"apiVersion":"hive.openshift.io/v1","kind":"Cl|...
```

無効な要求を削除できます。

不正な要求が削除されると、要求は追加の対話なしに正常に調整を開始します。

1.1.2.1.6. 製品チャネルが、プロビジョニングされたクラスターと同期されない

clusterimageset は **fast** チャネルに置かれますが、プロビジョニングされたクラスターは **stable** チャネルにあります。現時点で、製品は **channel** をプロビジョニングされた OpenShift Container Platform クラスターと同期しません。

OpenShift Container Platform コンソールで適切なチャネルに切り替えます。**Administration** > **Cluster Settings** > **Details Channel** の順にクリックします。

1.1.2.1.7. カスタム CA 証明書を使用したマネージドクラスターの、復元されたハブクラスターへの接続の復元は失敗する可能性がある

カスタム CA 証明書を使用してクラスターを管理したハブクラスターのバックアップを復元した後、マネージドクラスターとハブクラスター間の接続が失敗する場合があります。これは、復元されたハブクラスターで CA 証明書がバックアップされなかったためです。接続を復元するには、マネージドクラスターの namespace にあるカスタム CA 証明書情報を、復元されたハブクラスターの **<managed_cluster>-admin-kubeconfig** シークレットにコピーします。

ヒント: バックアップコピーを作成する前にこの CA 証明書をハブクラスターにコピーする場合は、バックアップコピーにシークレット情報が含まれます。将来、バックアップコピーを使用して復元する場合、ハブとマネージドクラスター間の接続は自動的に完了します。

1.1.2.1.8. ローカルクラスターが自動的に再作成されない場合がある

disableHubSelfManagement が **false** に設定されている場合、local-cluster は **MulticlusterHub** Operator によって再作成されます。ローカルクラスターをデタッチした後、ローカルクラスターが自動的に再作成されない場合があります。

- この問題を解決するには、**MulticlusterHub** によって監視されるリソースを変更します。以下の例を参照してください。

```
oc delete deployment multiclusterhub-repo -n <namespace>
```

- local-cluster を適切にデタッチするには、**MultiClusterHub** で **disableHubSelfManagement** を true に設定します。

1.1.2.1.9. オンプレミスクラスターを作成する場合は、サブネットを選択する必要がある

コンソールを使用してオンプレミスクラスターを作成する場合は、クラスターで利用可能なサブネットを選択する必要があります。必須フィールドとしてマークされていません。

1.1.2.1.10. Infrastructure Operator を使用したクラスターのプロビジョニングに失敗する

Infrastructure Operator を使用して OpenShift Container Platform クラスターを作成する場合、ISO イメージのファイル名は長すぎる可能性があります。長いイメージ名により、イメージのプロビジョニングとクラスターのプロビジョニングが失敗します。この問題が生じるかどうかを確認するには、以下の手順を実行します。

1. 以下のコマンドを実行して、プロビジョニングするクラスターのベアメタルホスト情報を表示します。

```
oc get bmh -n <cluster_provisioning_namespace>
```

2. **describe** コマンドを実行して、エラー情報を表示します。

```
oc describe bmh -n <cluster_provisioning_namespace> <bmh_name>
```

3. 以下の例と同様のエラーは、ファイル名の長さが問題であることを示します。

```
Status:
  Error Count:    1
  Error Message:  Image provisioning failed: ... [Errno 36] File name too long ...
```

この問題が発生する場合、これは通常 OpenShift Container Platform の以下のバージョンで発生します。インフラストラクチャー Operator がイメージサービスを使用していないためです。

- 4.8.17 以前
- 4.9.6 以前

このエラーを回避するには、OpenShift Container Platform をバージョン 4.8.18 以降、または 4.9.7 以降にアップグレードしてください。

1.1.2.1.11. 別の名前で再インポートした後に local-cluster のステータスがオフラインになる

local-cluster という名前のクラスターを、誤って別の名前のクラスターとして再インポートしようとすると、**local-cluster** と再インポートしたクラスターのステータスが **offline** と表示されます。

このケースから回復するには、以下の手順を行います。

1. ハブクラスターで以下のコマンドを実行して、ハブクラスターの自己管理の設定を一時的に編集します。

```
oc edit mch -n open-cluster-management multiclusterhub
```

2. **spec.disableSelfManagement=true** の設定を追加します。
3. ハブクラスターで以下のコマンドを実行し、local-cluster を削除し、再デプロイします。

```
oc delete managedcluster local-cluster
```

4. 以下のコマンドを実行して **local-cluster** 管理設定を削除します。

```
oc edit mch -n open-cluster-management multiclusterhub
```

5. 前の手順で追加した **spec.disableSelfManagement=true** を削除します。

1.1.2.1.12. Ansible 自動化を使用したクラスタープロビジョニングがプロキシー環境で失敗する

マネージドクラスターを自動的にプロビジョニングするように設定された自動化テンプレートは、次の両方の条件が満たされた場合に失敗する可能性があります。

- ハブクラスターで、クラスター全体のプロキシーが有効になっている。
- Ansible Automation Platform には、プロキシー経由でのみアクセスできます。

1.1.2.1.13. klusterlet Operator のバージョンは、ハブクラスターと同じである必要がある

klusterlet Operator をインストールしてマネージドクラスターをインポートする場合には、klusterlet Operator のバージョンは、ハブクラスターのバージョンと同じでなければなりません。そうでないと、klusterlet Operator は動作しません。

1.1.2.1.14. マネージドクラスター namespace を手動で削除できない

マネージドクラスターの namespace を手動で削除できません。マネージドクラスター namespace は、マネージドクラスターの割り当てを解除した後に自動的に削除されます。マネージドクラスターの割り当てを解除する前に手動でマネージドクラスター namespace を削除する場合は、マネージドクラスターの削除後にマネージドクラスターに継続的な終了ステータスが表示されます。この終了マネージドクラスターを削除するには、割り当てを解除したマネージドクラスターからファイナライザーを手動で削除します。

1.1.2.1.15. ハブクラスターとマネージドクラスターのクロックが同期されない

ハブクラスターおよびマネージドクラスターの時間が同期されず、コンソールで **unknown** と表示され、最数的に、数分以内に **available** と表示されます。OpenShift Container Platform ハブクラスターの時間が正しく設定されていることを確認します。ノードのカスタマイズ を参照してください。

1.1.2.1.16. IBM OpenShift Container Platform Kubernetes Service クラスターの特定のバージョンのインポートはサポートされていない

IBM OpenShift Container Platform Kubernetes Service バージョン 3.11 のクラスターをインポートすることはできません。IBM OpenShift Kubernetes Service の 3.11 よりも後のバージョンはサポート対象です。

1.1.2.1.17. プロビジョニングされたクラスターのシークレットの自動更新はサポートされていない

クラウドプロバイダー側でクラウドプロバイダーのアクセスキーを変更する場合は、multicluster engine operator のコンソールでこのクラウドプロバイダーの対応する認証情報を更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

1.1.2.1.18. マネージドクラスターからのノード情報を検索で表示できない

検索で、ハブクラスターのリソース用の RBAC がマッピングされます。ユーザー RBAC の設定によっては、マネージドクラスターからのノードデータが表示されない場合があります。また検索の結果は、クラスターの **Nodes** ページに表示される内容と異なる場合があります。

1.1.2.1.19. クラスターを破棄するプロセスが完了しない

マネージドクラスターを破棄してから 1 時間経過してもステータスが **Destroying** のままで、クラスターが破棄されません。この問題を解決するには、以下の手順を実行します。

1. クラウドに孤立したリソースがなく、マネージドクラスターに関連付けられたプロバイダーリソースがすべて消去されていることを確認します。
2. 以下のコマンドを入力して、削除するマネージドクラスターの **ClusterDeployment** 情報を開きます。

   ```
   oc edit clusterdeployment/<mycluster> -n <namespace>
   ```

   mycluster は、破棄するマネージドクラスターの名前に置き換えます。

   namespace は、マネージドクラスターの namespace に置き換えます。
3. **hive.openshift.io/deprovision** ファイナライザーを削除し、クラウドのクラスターリソースを消去しようとするプロセスを強制的に停止します。

4. 変更を保存して、**ClusterDeployment** が削除されていることを確認します。
5. 以下のコマンドを実行してマネージドクラスターの namespace を手動で削除します。

```
oc delete ns <namespace>
```

namespace は、マネージドクラスターの namespace に置き換えます。

1.1.2.1.20. OpenShift Container Platform Dedicated でコンソールを使用して OpenShift Container Platform マネージドクラスターをアップグレードできない

Red Hat Advanced Cluster Management コンソールを使用して、OpenShift Container Platform Dedicated 環境にある OpenShift Container Platform マネージドクラスターをアップグレードすることはできません。

1.1.2.1.21. ワークマネージャーのアドオン検索の詳細

特定のマネージドクラスターにある特定のリソースの検索詳細ページで問題が発生する可能性があります。マネージドクラスターの work-manager アドオンが **Available** ステータスであることを確認してから検索する必要があります。

1.1.2.1.22. Red Hat OpenShift Container Platform 以外のマネージドクラスターでは、アップグレード後に Pod ログ用に ManagedServiceAccount または LoadBalancer が必要となる

Red Hat Advanced Cluster Management 2.10 以降の新規インストールを使用している場合、Red Hat OpenShift Container Platform クラスターと非 OpenShift Container Platform クラスターの両方が Pod ログ機能をサポートしています。

Red Hat Advanced Cluster Management 2.9 から 2.10 にアップグレードした場合、OpenShift Container Platform 以外のマネージドクラスターで Pod ログ機能を使用するには、**ManagedServiceAccount** アドオンを手動で有効にする必要があります。**ManagedServiceAccount** を有効にする方法については、ManagedServiceAccount アドオン を参照してください。

または、**ManagedServiceAccount** の代わりに **LoadBalancer** を使用して、OpenShift Container Platform 以外のマネージドクラスターで Pod ログ機能を有効にすることもできます。

LoadBalancer を有効にするには、以下の手順を実行します。

1. クラウドプロバイダーごとに **LoadBalancer** 設定が異なります。詳細は、クラウドプロバイダーのドキュメントを参照してください。
2. **managedClusterInfo** のステータスで **loggingEndpoint** をチェックして、**LoadBalancer** が Red Hat Advanced Cluster Management で有効にされているかどうかを確認します。
3. 以下のコマンドを実行して、**loggingEndpoint.IP** または **loggingEndpoint.Host** に有効な IP アドレスまたはホスト名が設定されていることを確認します。

```
oc get managedclusterinfo <clusterName> -n <clusterNamespace> -o json | jq -r
'.status.loggingEndpoint'
```

LoadBalancer のタイプについての詳細は、Kubernetes のドキュメント のService ページを参照してください。

1.1.2.1.23. OpenShift Container Platform 4.10.z では、プロキシー設定を使用する Hosted control plane クラスターはサポートされません

OpenShift Container Platform 4.10.z でクラスター全体のプロキシー設定を使用してホスティングサービスクラスターを作成すると、**nodeip-configuration.service** サービスがワーカーノードで開始されません。

1.1.2.1.24. Azure で OpenShift Container Platform 4.11 クラスターをプロビジョニングできない

Azure で OpenShift Container Platform 4.11 クラスターをプロビジョニングすると、認証 Operator のタイムアウトエラーが原因で失敗します。この問題を回避するには、**install-config.yaml** ファイルで別のワーカーノードタイプを使用するか、**vmNetworkingType** パラメーターを **Basic** に設定します。次の **install-config.yaml** の例を参照してください。

```
compute:
- hyperthreading: Enabled
  name: 'worker'
  replicas: 3
  platform:
    azure:
      type:  Standard_D2s_v3
      osDisk:
        diskSizeGB: 128
      vmNetworkingType: 'Basic'
```

1.1.2.1.25. クライアントが iPXE スクリプトにアクセスできない

iPXE は、オープンソースのネットワークブートファームウェアです。詳細は、iPXE を参照してください。

ノードの起動時に、一部の DHCP サーバーの URL の長さ制限により、**InfraEnv** カスタムリソース定義の **ipxeScript** URL が切り取られ、コンソールに次のエラーメッセージが表示されます。

起動可能なデバイスがありません

この問題を回避するには、以下の手順を実行します。

1. 自動インストールを使用して **bootArtifacts** を公開する場合は、**InfraEnv** カスタムリソース定義を適用します。これは次のファイルのようになります。

```
status:
  agentLabelSelector:
    matchLabels:
      infraenvs.agent-install.openshift.io: qe2
  bootArtifacts:
    initrd: https://assisted-image-service-multicluster-engine.redhat.com/images/0000/pxe-initrd?api_key=0000000&arch=x86_64&version=4.11
    ipxeScript: https://assisted-service-multicluster-engine.redhat.com/api/assisted-install/v2/infra-envs/00000/downloads/files?api_key=000000000&file_name=ipxe-script
    kernel: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.12/latest/rhcos-live-kernel-x86_64
    rootfs: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.12/latest/rhcos-live-rootfs.x86_64.img
```

2. 短い URL で **bootArtifacts** を公開するプロキシーサーバーを作成します。

3. 次のコマンドを実行して、**bootArtifacts** をコピーし、プロキシーに追加します。

```
for artifact in oc get infraenv qe2 -ojsonpath="{.status.bootArtifacts}" | jq ". | keys[]" | sed
"s/\"//g"
do curl -k oc get infraenv qe2 -ojsonpath="{.status.bootArtifacts.${artifact}}"` -o $artifact
```

4. **ipxeScript** アーティファクトプロキシー URL を **libvirt.xml** の **bootp** パラメーターに追加します。

1.1.2.1.26. Red Hat Advanced Cluster Management のアップグレード後にClusterDeployment を削除できない

Red Hat Advanced Cluster Management 2.6 で削除された BareMetalAssets API を使用している場合、BareMetalAssets API が **ClusterDeployment** にバインドされているため、Red Hat Advanced Cluster Management 2.7 にアップグレードした後に **ClusterDeployment** を削除することはできません。

この問題を回避するには、以下のコマンドを実行して、Red Hat Advanced Cluster Management 2.7 にアップグレードする前に **finalizers** を削除します。

```
oc patch clusterdeployment <clusterdeployment-name> -p '{"metadata":{"finalizers":null}}' --
type=merge
```

1.1.2.1.27. central infrastructure management サービスを使用して非接続環境にデプロイされたクラスターがインストールされない場合がある

Central Infrastructure Management サービスを使用して非接続環境でクラスターをデプロイすると、クラスターノードのインストールが開始されない場合があります。

この問題は、OpenShift Container Platform バージョン 4.12.0 から 4.12.2 に同梱されている Red Hat Enterprise Linux CoreOS ライブ ISO イメージから作成された検出 ISO イメージをクラスターが使用するために発生します。イメージには、**registry.redhat.io** および **registry.access.redhat.com** から取得したイメージの署名を必要とする制限付きの **/etc/containers/policy.json** ファイルが含まれています。非接続環境では、ミラーリングされたイメージにミラーリングされた署名がない場合があり、その結果、クラスターノードの検出時にイメージのプルが失敗します。エージェントイメージがクラスターノードとの接続に失敗するため、Assisted Service との通信が失敗します。

この問題を回避するには、**/etc/containers/policy.json** ファイルを制限なしに設定する ignition オーバーライドをクラスターに適用します。ignition オーバーライドは、**InfraEnv** カスタムリソース定義で設定できます。次の例は、オーバーライドを使用した **InfraEnv** カスタムリソース定義を示しています。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: cluster
  namespace: cluster
spec:
  ignitionConfigOverride: '{"ignition":{"version":"3.2.0"},"storage":{"files":
[{"path":"/etc/containers/policy.json","mode":420,"overwrite":true,"contents":
{"source":"data:text/plain;charset=utf-
8;base64,ewogICAgImRlZmF1bHQiOiBbCiAgICAgICAgewogICAgICAgICAgICAidHlwZSI6ICJpbnNlY3
VyZUFjY2VwdEFueXRoaW5nIgogICAgICAgIH0KICAgIF0sCiAgICAidHJhbnNwb3J0cyI6CiAgICAgICAg
ewogICAgICAgICAgICAiZG9ja2VyLWRhZW1vbiI6CiAgICAgICAgICAgICAgICAgICB7CiAgICAgICAgICAg
ICAgICAgICAgIiI6IFt7InR5cGUiOiJpbnNlY3VyZUFjY2VwdEFueXRoaW5nIn1dCiAgICAgICAgICAgIC
B9CiAgICAgICAgfQp9"}}]}}'
```

次の例は、作成される制限なしのファイルを示しています。

```
{
    "default": [
        {
            "type": "insecureAcceptAnything"
        }
    ],
    "transports": {
        "docker-daemon": {
        "": [
        {
            "type": "insecureAcceptAnything"
        }
        ]
    }
    }
}
```

この設定を変更すると、クラスターがインストールされます。

1.1.2.1.28. マネージドクラスターがデプロイ後に Pending ステータスのままになる

Assisted Installer エージェントの起動が遅く、マネージドクラスターをデプロイすると、マネージドクラスターが **Pending** ステータスのままになり、エージェントリソースがなくなる可能性があります。この問題は、統合フローを無効にすることで回避できます。以下の手順を実行します。

1. ハブクラスター上に次の ConfigMap を作成します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-assisted-service-config
  namespace: multicluster-engine
data:
  ALLOW_CONVERGED_FLOW: "false"
```

2. 次のコマンドを実行して、ConfigMap を適用します。

```
oc annotate --overwrite AgentServiceConfig agent unsupported.agent-
install.openshift.io/assisted-service-configmap=my-assisted-service-config
```

1.1.2.1.29. ManagedClusterSet API 仕様の制限

Clustersets API を使用する場合、**selectorType: LaberSelector** 設定がサポートされません。**selectorType: ExclusiveClusterSetLabel** 設定がサポートされています。

1.1.2.1.30. ハブクラスター通信の制限

ハブクラスターがマネージドクラスターにアクセスできない、またはマネージドクラスターと通信できない場合、次の制限が発生します。

- コンソールを使用して新しいマネージドクラスターを作成できません。コマンドラインインター

- コンソールを使用して新しいマネージドクラスターを作成できません。コマンドラインインターフェイスを使用するか、コンソールで **Run import commands manually** オプションを使用して、マネージドクラスターを手動でインポートできます。

- コンソールを使用してアプリケーションまたはアプリケーションセットをデプロイする場合、またはマネージドクラスターを ArgoCD にインポートする場合、ハブクラスター ArgoCD コントローラーはマネージドクラスター API サーバーを呼び出します。AppSub または ArgoCD pull モデルを使用して問題を回避できます。

- Pod ログのコンソールページは機能せず、以下のようなエラーメッセージが表示されます。

  ```
  Error querying resource logs:
  Service unavailable
  ```

1.1.2.1.31. Managed Service Account アドオンの制限

managed-serviceaccount アドオンの既知の問題と制限事項は次のとおりです。

1.1.2.1.31.1. installNamespace フィールドには値を 1 つだけ指定できる

managed-serviceaccount アドオンを有効にする場合、**ManagedClusterAddOn** リソースの **installNamespace** フィールドの値として **open-cluster-management-agent-addon** が必要です。その他の値は無視されます。**managed-serviceaccount** アドオンエージェントは、マネージドクラスターの **open-cluster-management-agent-addon** namespace に常にデプロイされます。

1.1.2.1.31.2. マネージドサービスアカウント エージェントは tolerations と nodeSelector の設定による影響を受けない

MultiClusterEngine および **MultiClusterHub** リソースに設定された **tolerations** と **nodeSelector** 設定は、ローカルクラスターにデプロイされた **managed-serviceaccount** エージェントには影響しません。**マネージドサービスアカウント** アドオンは、ローカルクラスターでは必ずしも必要というわけではありません。

managed-serviceaccount アドオンが必要な場合は、次の手順を実行することで問題を回避できます。

1. **addonDeploymentConfig** カスタムリソースを作成します。
2. ローカルクラスターおよび **managed-serviceaccount** エージェントの **tolerations** および **nodeSelector** の値を設定します。
3. 作成した **addonDeploymentConfig** カスタムリソースを使用するように、ローカルクラスター namespace で **managed-serviceaccount ManagedClusterAddon** を更新します。

addonDeploymentConfig カスタムリソースを使用してアドオンの **tolerations** と **nodeSelector** を設定する方法の詳細は、klusterlet アドオン の nodeSelectors と tolerations の設定を参照してください。

1.1.2.1.32. KubeVirt ホステッドクラスターの一括破棄オプションでホステッドクラスターが破棄されない

KubeVirt ホステッドクラスターのコンソールで一括破棄オプションを使用しても、KubeVirt ホステッドクラスターが破棄されません。

代わりに、行のアクションドロップダウンメニューを使用して、KubeVirt ホステッドクラスターを破棄してください。

1.1.2.1.33. クラスターキュレーターが OpenShift Container Platform Dedicated クラスターをサポートしていない

ClusterCurator リソースを使用して OpenShift Container Platform Dedicated クラスターをアップグレードすると、クラスターキュレーターが OpenShift Container Platform Dedicated クラスターをサポートしていないため、アップグレードが失敗します。

1.1.2.2. Hosted Control Plane

1.1.2.2.1. コンソールにホステッドクラスターが Pending import として表示される

アノテーションと **ManagedCluster** 名が一致しない場合、コンソールはクラスターを **Pending import** と表示します。クラスターは multicluster engine operator では使用できません。アノテーションがなく、**ManagedCluster** 名が **HostedCluster** リソースの **Infra-ID** 値と一致しない場合は、同じ問題が発生します。

1.1.2.2.2. コンソールは、ホステッドクラスターにノードプールを追加する際に、同じバージョンを複数回、一覧表示する場合があります。

コンソールを使用して既存のホステッドクラスターに新規ノードプールを追加すると、同じバージョンの OpenShift Container Platform がオプションの一覧に複数回、表示される可能性があります。必要なバージョンの一覧で任意のインスタンスを選択できます。

1.1.2.2.3. カスタム Ingress ドメインが正しく適用されない

マネージドクラスターのインストール中に **ClusterDeployment** リソースを使用してカスタム Ingress ドメインを指定できますが、変更はインストール後に **SyncSet** リソースを使用してのみ適用されます。その結果、**clusterdeployment.yaml** ファイルの **spec** フィールドには、指定したカスタム Ingress ドメインが表示されますが、**status** には引き続きデフォルトのドメインが表示されます。

1.1.2.2.4. Web コンソールには、ノードがクラスターから削除されインフラストラクチャー環境に戻された後でもノードがリストされます。

ノードプールが 0 ワーカーにスケールダウンされても、コンソールのホストのリストには、**Ready** 状態のノードが表示されます。ノードの数は、次の 2 つの方法で確認できます。

- コンソールでノードプールに移動し、ノードが 0 であることを確認します。
- コマンドラインインターフェイスで、以下のコマンドを実行します。
  - 次のコマンドを実行して、ノードプールにあるノード数が 0 個であることを確認します。

    ```
    oc get nodepool -A
    ```

  - 次のコマンドを実行して、クラスター内にあるノード数が 0 個であることを確認します。

    ```
    oc get nodes --kubeconfig
    ```

  - 次のコマンドを実行して、クラスターにバインドされているエージェント数が 0 と報告されていることを確認します。

```
oc get agents -A
```

1.1.2.2.5. デュアルスタックネットワーク用に設定されたホステッドクラスターで DNS の問題が発生する可能性がある

デュアルスタックネットワークを使用する環境でホステッドクラスターを作成すると、次の DNS 関連の問題が発生する可能性があります。

- **Service-ca-operator** Pod の **CrashLoopBackOff** 状態: Pod が Hosted control plane 経由で Kubernetes API サーバーに到達しようとすると、**kube-system** namespace のデータプレーンプロキシーがリクエストを解決できないため、Pod はサーバーに到達できません。この問題は、HAProxy セットアップでフロントエンドが IP アドレスを使用し、バックエンドが Pod が解決できない DNS 名を使用するために発生します。

- Pod が **ContainerCreating** 状態でスタックする: この問題は、**openshift-service-ca-operator** が DNS Pod が DNS 解決に必要とする **metrics-tls** シークレットを生成できないために発生します。その結果、Pod は Kubernetes API サーバーを解決できません。

これらの問題を解決するには、デュアルスタックネットワーク用の DNS の設定 のガイドラインに従って DNS サーバー設定を指定します。

1.1.2.2.6. ベアメタルプラットフォームでは、エージェントリソースが Ignition に失敗することがある

ベアメタル (エージェント) プラットフォームでは、Hosted control plane 機能により、エージェントがイグニションのプルに使用するトークンが定期的にローテーションされます。バグにより、新しいトークンが伝播されません。その結果、少し前に作成されたエージェントリソースがある場合、Ignition のプルに失敗する可能性があります。

回避策として、エージェント仕様で、**IgnitionEndpointTokenReference** プロパティーが参照するシークレットを削除し、エージェントリソースのラベルを追加または変更します。その後、システムはエージェントリソースが変更されたことを検出し、新しいトークンを使用してシークレットを再作成できます。

1.1.3. エラータの更新

multicluster engine operator の場合、エラータの更新はリリース時に自動的に適用されます。

重要: 参照できるように、エラータ リンクと GitHub 番号がコンテンツに追加され、内部で使用される可能性があります。ユーザーは、アクセス権が必要なリンクを利用できない可能性があります。

1.1.3.1. エラータ 2.5.3

- KubeVirt 作成ウィザードに、Hosted Control Plane クラスターのデフォルトモードを **HighAvailability** モードに設定するフィールドが追加されました。(ACM-10580)

- 1 つ以上の製品コンテナーイメージに更新を配信します。

1.1.3.2. エラータ 2.5.2

- バックアップ/復元シナリオを実行し、Red Hat OpenShift Data Foundation (ODF) の Regional-DR ソリューションを使用すると、データ損失が発生する可能性がある問題を修正しました。(ACM-10407)

- 1 つ以上の製品コンテナーイメージに更新を配信します。

1.1.3.3. エラータ 2.5.1

- 1 つ以上の製品コンテナーイメージに更新を配信します。

1.1.4. 非推奨とクラスターライフサイクルの削除

製品の一部が非推奨または multicluster engine operator から削除されるタイミングを確認します。**推奨アクション** および詳細にある、代わりのアクションを検討してください。これについては、現在のリリースおよび、1 つ前のリリースと 2 つ前のリリースの表に記載されています。

1.1.4.1. API の非推奨と削除

multicluster engine operator は、Kubernetes の API 非推奨ガイドラインに従います。そのポリシーに関する詳細は、Kubernetes の非推奨ポリシー を参照してください。multicluster engine Operator API は、以下のタイムライン外でのみ非推奨または削除されます。

- **V1** API はすべて、12 ヶ月間または リリース 3 回分 (いずれか長い方) の期間は一般公開され、サポート対象となります。V1 API は削除されませんが、この期間を過ぎると非推奨になる可能性があります。
- **Beta** 版 API はすべて、9ヶ月間またはリリース 3 回分 (いずれか長い方) の期間は一般公開されます。Beta 版 API は、この期間を過ぎても削除されません。
- **Alpha** 版 API はサポートの必要はありませんが、ユーザーにとってメリットがある場合には、非推奨または削除予定として記載される場合があります。

1.1.4.1.1. API の非推奨化

製品またはカテゴリー	影響を受けるアイテム	バージョン	推奨されるアクション	詳細およびリンク
ManagedServiceAccount	**v1alpha1** は非推奨となったため、**v1alpha1** API は **v1beta1** にアップグレードされます。	2.9	**V1beta1** を使用してください。	なし

1.1.4.1.2. API の削除

製品またはカテゴリー	影響を受けるアイテム	バージョン	推奨されるアクション	詳細およびリンク

1.1.4.2. 非推奨

非推奨 のコンポーネント、機能またはサービスはサポートされますが、使用は推奨されておらず、今後のリリースで廃止される可能性があります。以下の表に記載されている **推奨アクション** と詳細の代替アクションについて検討してください。

製品またはカテゴリー	影響を受けるアイテム	バージョン	推奨されるアクション	詳細およびリンク
クラスターライフサイクル	Red Hat Virtualization でのクラスターの作成	2.9	なし	なし

製品またはカテゴリー	影響を受けるアイテム	バージョン	推奨されるアクション	詳細およびリンク
クラスターライフサイクル	klusterlet OLM Operator	2.4	なし	なし

1.1.4.3. 削除

通常、**削除** された項目は、以前のリリースで非推奨となった機能で、製品では利用できなくなっています。削除された機能には、代わりの方法を使用する必要があります。以下の表に記載されている **推奨アクション** と詳細の代替アクションについて検討してください。

製品またはカテゴリー	影響を受けるアイテム	バージョン	推奨されるアクション	詳細およびリンク

1.2. MULTICLUSTER ENGINE OPERATOR を使用したクラスターライフサイクルについて

Kubernetes Operator のマルチクラスターエンジンは、Red Hat OpenShift Container Platform および Red Hat Advanced Cluster Management ハブクラスターにクラスター管理機能を提供するクラスターライフサイクル Operator です。Red Hat Advanced Cluster Management をインストールした場合は、自動的にインストールされるため、multicluster engine operator をインストールする必要はありません。

ハブクラスター、マネージドクラスターの要件およびサポート情報については、サポートマトリックスと以下のドキュメントを参照してください。

- コンソールの概要
- Kubernetes Operator のロールベースのアクセス制御用のマルチクラスターエンジン
- ネットワーク設定

続行するには、multicluster engine Operator を使用したクラスターライフサイクルについて で、残りのクラスターライフスタイルドキュメントを参照してください。

1.2.1. 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 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場合、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、クラスタースイッチャーから OpenShift Container Platform コンソール内の 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** プラグインがリスト表示されます。**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。

1.2.2. multicluster engine operator のロールベースのアクセス制御

RBAC はコンソールレベルと API レベルで検証されます。コンソール内のアクションは、ユーザーのアクセスロールの権限に基づいて有効化/無効化できます。製品の特定ライフサイクルの RBAC の詳細は、以下のセクションを参照してください。

- ロールの概要
- クラスターライフサイクル RBAC
  - クラスタープール RBAC
  - クラスターライフサイクルのコンソールおよび API RBAC の表
  - 認証情報ロールベースのアクセス制御

1.2.2.1. ロールの概要

クラスター別の製品リソースと、スコープに namespace が指定されている製品リソースがあります。アクセス制御に一貫性を持たせるため、クラスターのロールバインドと、namespace のロールバインドをユーザーに適用する必要があります。サポートされている次のロール定義の表リストを表示します。

1.2.2.1.1. ロール定義表

ロール	定義
cluster-admin	これは OpenShift Container Platform のデフォルトのロールです。**cluster-admin** ロールへのクラスターバインディングがあるユーザーは、すべてのアクセス権限を持つ OpenShift Container Platform のスーパーユーザーです。
open-cluster-management:cluster-manager-admin	**open-cluster-management:cluster-manager-admin** ロールにクラスターをバインドするユーザーは、すべてのアクセス権を持つスーパーユーザーです。このロールを指定すると、ユーザーは **ManagedCluster** リソースを作成できます。

ロール	定義
open-cluster-management:admin:<managed_cluster_name>	**open-cluster-management:admin:<managed_cluster_name>** ロールへのクラスターバインディングがあるユーザーには、**managedcluster-<managed_cluster_name>** という名前の **ManagedCluster** リソースに管理者アクセス権が付与されます。ユーザーにマネージドクラスターがある場合は、このロールが自動的に作成されます。
open-cluster-management:view:<managed_cluster_name>	**open-cluster-management:view:<managed_cluster_name>** ロールへのクラスターバインディングがあるユーザーには、**managedcluster-<managed_cluster_name>** という名前の **ManagedCluster** リソースの表示権限が付与されます。
open-cluster-management:managedclusterset:admin:<managed_clusterset_name>	**open-cluster-management:managedclusterset:admin:<managed_clusterset_name>** ロールへのクラスターバインドのあるユーザーには、**<managed_clusterset_name>** という名前の **ManagedCluster** リソースへの管理者アクセスがあります。また、ユーザーには **managedcluster.cluster.open-cluster-management.io**、**clusterclaim.hive.openshift.io**、**clusterdeployment.hive.openshift.io** および **clusterpool.hive.openshift.io** リソースへの管理者アクセスがあり、**cluster.open-cluster-management.io** と **clusterset=<managed_clusterset_name>** のマネージドクラスターセットのラベルが付いています。ロールバインディングは、クラスターセットの使用時に自動的に生成されます。リソースの管理方法については、ManagedClusterSet の作成 を参照してください。

ロール	定義
open-cluster-management:managedclusterset:view:<managed_clusterset_name>	**open-cluster-management:managedclusterset:view:<managed_clusterset_name>** ロールへのクラスターバインディングがあるユーザーには、<managed_clusterset_name>` という名前の **ManagedCluster** リソースへの表示権限が付与されます。また、ユーザーには **managedcluster.cluster.open-cluster-management.io**、**clusterclaim.hive.openshift.io**、**clusterdeployment.hive.openshift.io** および **clusterpool.hive.openshift.io** リソースの表示権限があります。これには、**cluster.open-cluster-management.io**、**clusterset=<managed_clusterset_name>** のマネージドクラスターセットのラベルが付いています。マネージドクラスターセットの管理方法の詳細は、ManagedClusterSet の作成 を参照してください。
admin、**edit**、**view**	admin、edit、および view は OpenShift Container Platform のデフォルトロールです。これらのロールに対して namespace に限定されたバインドが指定されているユーザーは、特定の namespace 内の **open-cluster-management** リソースにアクセスでき、同じロールに対してクラスター全体のバインドが指定されている場合には、クラスター全体の全 **open-cluster-management** リソースにアクセス権があります。

重要:

- ユーザーは OpenShift Container Platform からプロジェクトを作成できます。これにより、namespace の管理者ロール権限が付与されます。
- ユーザーにクラスターへのロールアクセスがない場合、クラスター名は表示されません。クラスター名は、**-** の記号で表示されます。

RBAC はコンソールレベルと API レベルで検証されます。コンソール内のアクションは、ユーザーのアクセスロールの権限に基づいて有効化/無効化できます。製品の特定ライフサイクルの RBAC の詳細は、以下のセクションを参照してください。

1.2.2.2. クラスターライフサイクル RBAC

以下のクラスターライフサイクル RBAC 操作を確認してください。

- すべてのマネージドクラスターのクラスターロールバインドを作成および管理します。たとえば、以下のコマンドを入力してクラスターロール **open-cluster-management:cluster-manager-admin** にバインドするクラスターロールを作成します。

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

このロールはスーパーユーザーであるため、すべてのリソースとアクションにアクセスできます。このロールを使用すると、クラスターレベルの **managedcluster** リソース、マネージドクラスターを管理するリソースの namespace、namespace 内のリソースを作成できます。権限エラーを回避するために、ロールの関連付けが必要な ID の **username** を追加する必要がある場合があります。

- 以下のコマンドを実行して、**cluster-name** という名前のマネージドクラスターのクラスターロールバインドを管理します。

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

このロールを使用すると、クラスターレベルの **managedcluster** リソースに読み取り/書き込みアクセスができるようになります。**managedcluster** はクラスターレベルのリソースで、namespace レベルのリソースではないので、このロールが必要です。

  - 以下のコマンドを入力して、クラスターロール **admin** にバインドする namespace ロールを作成します。

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

このロールでは、マネージドクラスターの namespace 内にあるリソースに対して読み取り/書き込みアクセスができるようになります。

- **open-cluster-management:view:<cluster-name>** クラスターロールのクラスターロールバインドを作成して、**cluster-name** という名前のマネージドクラスターを表示します。次のコマンドを入力します。

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

このロールを使用すると、クラスターレベルの **managedcluster** リソースに読み取りアクセスができるようになります。これは、**managedcluster** がクラスタースコープのリソースであるために必要です。

- 以下のコマンドを入力して、クラスターロール **view** にバインドする namespace ロールを作成します。

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

このロールでは、マネージドクラスターの namespace 内にあるリソースに対して読み取り専用アクセスができるようになります。

- 以下のコマンドを入力して、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。

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

このコマンドは、クラスター管理者権限なしで、管理者およびユーザーが使用できます。

- 以下のコマンドを入力して、アクセス可能なマネージドクラスターセットの一覧を表示します。

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

このコマンドは、クラスター管理者権限なしで、管理者およびユーザーが使用できます。

1.2.2.2.1. クラスタープール RBAC

以下のクラスタープール RBAC 操作を確認します。

- クラスター管理者は、クラスタープールのプロビジョニングクラスターを使用して、マネージドクラスターセットを作成し、ロールをグループに追加して管理者権限をロールに付与します。以下の例を参照してください。
  - 以下のコマンドを使用して、**server-foundation-clusterset** マネージドクラスターセットに **admin** 権限を付与します。

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

  - 以下のコマンドを使用して、**server-foundation-clusterset** マネージドクラスターセットに **view** 権限を付与します。

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

- クラスタープールの namespace (**server-foundation-clusterpool**) を作成します。ロール権限を付与するには、以下の例を参照してください。
  - 以下のコマンドを実行して、**server-foundation-team-admin** の **server-foundation-clusterpool** に **admin** 権限を付与します。

```
oc adm new-project server-foundation-clusterpool

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

- チーム管理者として、クラスタープール namespace にクラスターセットラベル **cluster.open-cluster-management.io/clusterset=server-foundation-clusterset** を使用して **ocp46-aws-clusterpool** という名前のクラスタープールを作成します。
  - **server-foundation-webhook** は、クラスタープールにクラスターセットラベルがあるかどうか、またユーザーにクラスターセットのクラスタープールを作成する権限があるかどうかを確認します。
  - **server-foundation-controller** は、**server-foundation-team-user** の **server-foundation-clusterpool** namespace に **view** 権限を付与します。
- クラスタープールが作成されると、クラスタープールは **clusterdeployment** を作成します。詳細は、以下を参照してください。
  - **server-foundation-controller** は、**server-foundation-team-admin** の **clusterdeployment** namespace に **admin** 権限を付与します。

- **server-foundation-controller** は、**server-foundation-team-user** の **clusterdeployment** namespace に **view** 権限を付与します。
  注記: team-admin および **team-user** には、**clusterpool**、**clusterdeployment**、および **clusterclaim** への **admin** 権限があります。

1.2.2.2.2. クラスターライフサイクルのコンソールおよび API RBAC の表

クラスターライフサイクルの以下のコンソールおよび API RBAC の表を表示します。

表1.1 クラスターライフサイクルのコンソール RBAC の表

リソース	管理	編集	表示
クラスター	read, update, delete	-	read
クラスターセット	get, update, bind, join	編集ロールなし	get
マネージドクラスター	read, update, delete	編集ロールなし	get
プロバイダー接続	create, read, update, delete	-	read

表1.2 クラスターライフサイクルの API RBAC の表

API	管理	編集	表示
managedclusters.cluster.open-cluster-management.io **この API のコマンドでは、mcl (単数) または mcls (複数) を使用できます。**	create, read, update, delete	read, update	read
managedclusters.view.open-cluster-management.io **この API のコマンドでは、mcv (単数) または mcvs (複数) を使用できます。**	read	read	read
managedclusters.register.open-cluster-management.io/accept	update	update	

API	管理	編集	表示
managedclusterset.cluster.open-cluster-management.io **この API のコマンドでは、mclset (単数) または mclsets (複数) を使用できます。**	create, read, update, delete	read, update	read
managedclustersets.view.open-cluster-management.io	read	read	read
managedclustersetbinding.cluster.open-cluster-management.io **この API のコマンドでは、mclsetbinding (単数) または mclsetbindings (複数) を使用できます。**	create, read, update, delete	read, update	read
klusterletaddonconfigs.agent.open-cluster-management.io	create, read, update, delete	read, update	read
managedclusteractions.action.open-cluster-management.io	create, read, update, delete	read, update	read
managedclusterviews.view.open-cluster-management.io	create, read, update, delete	read, update	read
managedclusterinfos.internal.open-cluster-management.io	create, read, update, delete	read, update	read
manifestworks.work.open-cluster-management.io	create, read, update, delete	read, update	read

API	管理	編集	表示
submarinerconfigs.submarineraddon.open-cluster-management.io	create, read, update, delete	read, update	read
placements.cluster.open-cluster-management.io	create, read, update, delete	read, update	read

1.2.2.2.3. 認証情報ロールベースのアクセス制御

認証情報へのアクセスは Kubernetes で制御されます。認証情報は Kubernetes Secret として保存され、セキュリティーを確保します。以下の権限は、Red Hat Advanced Cluster Management for Kubernetes のシークレットのアクセスに関係します。

- namespace でシークレットの作成権限のあるユーザーは認証情報を作成できます。
- namespace でシークレットの読み取り権限のあるユーザーは、認証情報を表示することもできます。
- Kubernetes ロール **admin** および **edit** のあるユーザーは、シークレットの作成と編集が可能です。
- Kubernetes クラスターロール **view** のあるユーザーは、シークレットの内容を読み取ると、サービスアカウントの認証情報にアクセスできるようになるため、シークレットを表示できません。

1.2.3. ネットワーク設定

接続を許可するようにネットワーク設定を設定します。

重要: 信頼できる CA バンドルは multicluster engine Operator namespace で利用できますが、その拡張にはネットワークへの変更が必要です。信頼できる CA バンドル ConfigMap は、**trusted-ca-bundle** のデフォルト名を使用します。この名前は、**TRUSTED_CA_BUNDLE** という名前の環境変数で Operator に提供すると変更できます。詳細は、Red Hat OpenShift Container Platform の **ネットワーク** セクションの クラスター全体のプロキシーの設定 を参照してください。

注記: マネージドクラスターの **Registration Agent** および **Work Agent** は、プロキシーを通過できない mTLS 接続の確立によりハブクラスターの **apiserver** と通信するため、プロキシー設定をサポートしません。

multicluster engine Operator のクラスターネットワーク要件については、次の表を参照してください。

方向	プロトコル	接続	ポート (指定されている場合)
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

方向	プロトコル	接続	ポート (指定されている場合)
OpenShift Container Platform マネージドクラスターからハブクラスターへの送信	TCP	Ironic エージェントとハブクラスター上のベアメタルオペレーター間の通信	6180、6183、6385、5050
ハブクラスターからマネージドクラスターの Ironic Python Agent (IPA) への送信	TCP	IPA が実行されているベアメタルノードと Ironic conductor サービス間の通信	9999
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 Operator クラスター用マルチクラスターエンジンの Kubernetes API サーバー	6443

注記: マネージドクラスターは、ハブクラスターのコントロールプレーンノードの IP アドレスに到達できる必要があります。

1.3. MULTICLUSTER ENGINE OPERATOR のインストールとアップグレード

multicluster engine Operator は、クラスターフリート管理を強化するソフトウェア Operator です。multicluster engine Operator は、クラウドおよびデータセンター全体の Red Hat OpenShift Container Platform および Kubernetes クラスターライフサイクル管理をサポートします。

ハブクラスターとマネージドクラスターの要件とサポートについては、サポートマトリックス にアクセスしてください。

重要: バージョン 2.5 以降で Red Hat Advanced Cluster Management を使用している場合、Kubernetes Operator 用のマルチクラスターエンジンはすでにクラスターにインストールされています。

以下のドキュメントを参照してください。

- ネットワーク接続時のオンラインインストール * ネットワーク切断状態でのインストール
- アンインストール
- MultiClusterEngine の高度な設定
- Red Hat Advanced Cluster Management の統合

1.3.1. ネットワーク接続時のオンラインインストール

multicluster engine Operator は、multicluster engine Operator を含むコンポーネントのインストール、アップグレード、および削除を管理する Operator Lifecycle Manager でインストールされます。

必要なアクセス権限: クラスターの管理者

重要:

- 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権限がありません。
- デフォルトでは、multicluster engine Operator 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 multicluster engine Operator 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 multicluster engine Operator 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。詳細については、**インフラストラクチャーノードへの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 Operator 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マルチクラスターエンジンのインストール

1.3.1.1. 前提条件

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前に、次の要件を確認してください。

- Red Hat OpenShift Container Platform クラスターは、OpenShift Container Platform コンソールから OperatorHub カタログの multicluster engine Operator にアクセスできる必要があります。
- catalog.redhat.com へのアクセスがある。
- お使いの環境に OpenShift Container Platform 4.13 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 バージョン 4.13

- 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。
- Red Hat OpenShift Container Platform で提供される Assisted Installer を使用してマネージドクラスターを作成する予定の場合は、OpenShift Container Platform ドキュメントの アシステッドインストーラーを使用したインストールの準備 トピックを参照してください。

1.3.1.2. 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. multicluster engine Operator が OpenShift Container Platform クラスターにインストールされていないことを確認します。multicluster engine Operator は、各 OpenShift Container Platform クラスターで 1 つのインストールのみを許可します。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 console
   ```

   以下の出力例を参照してください。

   ```
   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。

multicluster engine operator のインストールに進むことができます。

1.3.1.3. 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:

     - multicluster engine Operator 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。multicluster engine Operator エンジンがビルドおよび起動するまで最長 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。

1.3.1.4. OpenShift Container Platform CLI からのインストール

1. Operator 要件を満たした multicluster engine Operator エンジン namespace を作成します。次のコマンドを実行します。ここで、**namespace** は、Kubernetes Operator 用マルチクラスターエンジンの namespace 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 Operator 用マルチクラスターエンジンの namespace の名前に置き換えます。

```
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>
  namespace: <namespace>
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-2.1
  installPlanApproval: Automatic
  name: multicluster-engine
```

注記: インフラストラクチャーノードに 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 multicluster engine Operator をインストールする場合は、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}'
```

multicluster engine Operator を再インストールし、Pod が起動しない場合は、この問題の回避手順について 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** が指定された **ServiceAccount** は、クラスター管理者権限を multicluster engine Operator と、multicluster engine Operator 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。

1.3.1.5. 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。

1.3.1.5.1. 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ってください。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。

multicluster engine operator が提供す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ます。

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

1.3.1.5.2. 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
```

1.3.1.5.3. 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: ""
```

1.3.2. 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 multicluster engine 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 multicluster engine Operator をインストールする必要があります。multicluster engine Operator 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。Red Hat Advanced Cluster Management をインストールしたことがないクラスターに multicluster engine Operator をインストールすることが推奨されます。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、multicluster engine Operator はすで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
- 非接続環境でのインストール

1.3.2.1. 前提条件

multicluster engine 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 4.13 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 にアクセスできる。
  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 4.13 以降が必要です。

  OpenShift Container Platform バージョン 4.13 を参照してください。
- Red Hat OpenShift Container Platform の CLI がバージョン 4.13 以降であり、**oc** コマンドを実行するように設定されている。
- 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。

1.3.2.2. 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 4.13 の詳細は、OpenShift Container Platform ドキュメント を参照してください。
- 接続されている場合は、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することにより、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。

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

  以下の出力例を参照してください。

  ```
  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。

1.3.2.3. 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。

以下の手順に従って、非接続環境に multicluster engine Operator 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まだない場合は、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。保護された非接続レジストリー内のイメージにアクセスするには、multicluster engine Operator がレジストリーにアクセスできるように、証明書情報を指定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: **registry.redhat.io/quay** ソースでイメージを見つけます。
   - Compliance Operator ポリシー: **registry.redhat.io/compliance** ソースでイメージを見つけます。
   - Gatekeeper Operator ポリシー: **registry.redhat.io/gatekeeper** ソースでイメージを見つけます。
     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. **mce-policy.yaml** という名前の **ImageContentSourcePolicy** を含む 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。
multicluster engine Operator は Operator Lifecycle Manager Red Hat Operator カタログに含まれます。

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. 非接続の Operator Lifecycle Manager にイメージを取得したので、引き続き Operator Lifecycle Manager カタログから Kubernetes 用の multicluster engine Operator 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 を参照してください。

1.3.3. 詳細設定

multicluster engine Operator 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。multicluster engine Operator は、インストール中またはインストール後にさらに設定できます。ここでは、詳細設定オプションについて説明します。

1.3.3.1. デプロイされるコンポーネント

次の属性を 1 つ以上 **MultiClusterEngine** カスタムリソースに追加します。

表1.3 デプロイされるコンポーネントのリストの表

名前	説明	有効
assisted-service	最小限のインフラストラクチャー前提条件と包括的なプリフライト検証を使用して OpenShift Container Platform をインストールします。	True

cluster-lifecycle	OpenShift Container Platform および Kubernetes ハブクラスターにクラスター管理機能を提供します。	True
cluster-manager	クラスター環境内のさまざまなクラスター関連操作を管理します。	True
cluster-proxy-addon	リバースプロキシーサーバーを使用して、ハブクラスターとマネージドクラスター両方での **apiserver-network-proxy** のインストールを自動化します。	True
console-mce	multicluster engine Operator コンソールのプラグインを有効にします。	True
discovery	OpenShift Cluster Manager 内の新しいクラスターを検出して識別します。	True
hive	OpenShift Container Platform クラスターの初期設定をプロビジョニングして実行します。	True
hypershift	コストと時間の効率性、クラウド間の移植性を備えた OpenShift Container Platform コントロールプレーンを大規模にホストします。	True
hypershift-local-hosting	ローカルクラスター環境内でのローカルホスティング機能を有効にします。	True
local-cluster	multicluster engine Operator がデプロイされているローカルハブクラスターのインポートと自己管理を有効にします。	True
managedserviceacccount	サービスアカウントをマネージドクラスターに同期し、トークンをシークレットリソースとして収集してハブクラスターに戻します。	False
server-foundation	マルチクラスター環境内のサーバー側操作のための基本的なサービスを提供します。	True

multicluster engine Operator をクラスターにインストールする場合、リストされているコンポーネントのすべてがデフォルトで有効になるわけではありません。

MultiClusterEngine カスタムリソースに 1 つ以上の属性を追加することで、インストール中またはインストール後に multicluster engine Operator をさらに設定できます。追加できる属性については、このまま読み進めてください。

1.3.3.2. コンソールとコンポーネントの設定

次の例では、コンポーネントを有効または無効にするために使用できる **spec.overrides** デフォルトテンプレートを表示します。

```
apiVersion: operator.open-cluster-management.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: <name> 1
      enabled: true
```

1. **name** をコンポーネントの名前に置き換えます。

あるいは、以下のコマンドを実行します。**namespace** プロジェクトの名前に置き換え、**name** をコンポーネントの名前に置き換えます。

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

1.3.3.3. ローカルクラスターの有効化

デフォルトでは、multicluster engine Operator を実行しているクラスターが自身を管理します。クラスター自身を管理せずに、multicluster engine Operator をインストールするには、**MultiClusterEngine** セクションの **spec.overrides.components** 設定で次の値を指定します。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: local-cluster
      enabled: false
```

- **name** 値は、ハブクラスターを **local-cluster** として識別します。
- **enabled** 設定は、機能を有効にするか無効にするかを指定します。値が **true** の場合、ハブクラスターは自身を管理します。値が **false** の場合、ハブクラスターは自身を管理しません。

自己管理されるハブクラスターは、クラスターの一覧で **local-cluster** として指定されます。

1.3.3.4. カスタムイメージプルシークレット

OpenShift Container Platform または multicluster engine operator 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予定の場合は、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 Kubernetes Operator 用のマルチクラスターエンジン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>
```

1.3.3.5. 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>
```

1.3.3.6. availabilityConfig

ハブクラスターには、**High** と **Basic** の 2 つの可用性があります。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。

重要: シングルノード OpenShift クラスターで multicluster engine Operator を使用している場合は、**spec.availabilityConfig** を **Basic** に設定します。

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"
```

1.3.3.7. 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。次の例は、ラベル **node-role.kubernetes.io/infra** を持つノードに Pod を割り当てる **spec.nodeSelector** を示しています。

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

1.3.3.8. Toleration

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。

1.3.3.9. ManagedServiceAccount アドオン

ManagedServiceAccount アドオンを使用すると、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
      enabled: true
```

ManagedServiceAccount 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** 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","enabled":true}}]'
```

1.3.4. アンインストール

Kubernetes Operator 用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。

1.3.4.1. 前提条件: 有効化されたサービスのデタッチ

Kubernetes Operator 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、クラスター作成の概要 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。

1.3.4.2. 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の詳細は、OpenShift Container Platform ドキュメントの OpenShift CLI スタートガイド 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。

1.3.4.3. 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: 同じバージョンのKubernetes Operator 用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、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。

1.3.4.4. 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
```

詳細は オフラインインストールのミラーリング を参照してください。

1.3.5. Red Hat Advanced Cluster Management の統合

Red Hat Advanced Cluster Management を有効にして multicluster engine Operator を使用している場合は、さらに多くのマルチクラスター管理機能を利用できます。

1.3.5.1. 前提条件

Red Hat Advanced Cluster Management 機能と統合するには、次の前提条件を参照してください。

- Red Hat Advanced Cluster Management がインストールされている。インストールするには、インストールとアップグレード を参照してください。

1.3.5.2. 可観測性の統合

multicluster-observability Pod を有効にすると、Red Hat Advanced Cluster Management Observability Grafana ダッシュボードを使用して、Hosted Control Plane に関する次の情報を表示できます。

- **ACM** > **Hosted Control Planes Overview** で、Hosted Control Plane をホストするためのクラスター容量の推定値、関連するクラスターリソース、および既存の Hosted Control Plane のリストとステータスを確認できます。
- **ACM** > **Resources** > **Hosted Control Plane** ダッシュボード (**Overview** ページからアクセス可能) で、選択した Hosted Control Plane のリソース使用率を確認できます。

有効にするには、可観測性サービスについて を参照してください。

1.4. 認証情報の管理

multicluster engine operator を使用してクラウドサービスプロバイダーで Red Hat OpenShift Container Platform クラスターを作成および管理するには、**認証情報** が必要です。認証情報では、クラウドプロバイダーのアクセス情報を保存します。1 つのプロバイダーのドメインごとに独自の認証情報が必要になるのと同様に、プロバイダーアカウントごとに独自の認証情報が必要です。

クラスターの認証情報を作成して管理できます。認証情報は Kubernetes Secret として保存されます。シークレットはマネージドクラスターの namespace にコピーされ、マネージドクラスターのコントローラーがシークレットにアクセスできるようになります。認証情報が更新されると、シークレットのコピーはマネージドクラスターの namespace で自動的に更新されます。

注記: クラウドプロバイダーの認証情報のプルシークレット、SSH キー、またはベースドメインへの変更は、元の認証情報を使用してすでにプロビジョニングされているため、既存のマネージドクラスターには反映されません。

必要なアクセス権限: 編集

- Amazon Web Services の認証情報の作成
- Microsoft Azure の認証情報の作成
- Google Cloud Platform の認証情報の作成
- VMware vSphere の認証情報の作成
- Red Hat OpenStack Platform の認証情報の作成
- Red Hat Virtualization の認証情報の作成
- Red Hat OpenShift Cluster Manager の認証情報の作成
- Ansible Automation Platform の認証情報の作成
- オンプレミス環境の認証情報の作成

1.4.1. Amazon Web Services の認証情報の作成

multicluster engine Operator コンソールを使用して、Amazon Web Services (AWS) で Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。

必要なアクセス権限: 編集

注記: この手順は、multicluster engine operator でクラスターを作成する前に実行する必要があります。

1.4.1.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- デプロイされた multicluster engine Operator ハブクラスター
- Amazon Web Services (AWS) で Kubernetes クラスターを作成できるようにする multicluster engine Operator ハブクラスターのインターネットアクセス
- アクセスキー ID およびシークレットアクセスキーなど、AWS のログイン認証情報。**Understanding and getting your security credentials**を参照してください。
- AWS でクラスターをインストールできるようにするアカウントの権限。**Configuring an AWS account** は、AWS アカウントの設定を参照してください。

1.4.1.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

オプションで、認証情報の **ベース DNS ドメイン** を追加できます。ベース DNS ドメインを認証情報に追加した場合は、この認証情報でクラスターを作成すると、このベース DNS ドメインは自動的に正しいフィールドに設定されます。以下の手順を参照してください。

1. AWS アカウントの **AWS アクセスキー ID** を追加します。ID を確認するには、 **Log in to AWS** を参照してください。
2. 新しい **AWS Secret Access Key** の内容を提供します。
3. プロキシーを有効にする必要がある場合は、プロキシー情報を入力します。
   - HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL。
   - HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
   - プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
   - 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。
4. Red Hat OpenShift プルシークレットを入力します。プルシークレットをダウンロードするには、**Download your Red Hat OpenShift pull secret**を参照してください。
5. **SSH 秘密鍵** と **SSH 公開鍵** を追加し、クラスターに接続できるようにします。既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。

Amazon Web Services でのクラスターの作成 または **Amazon Web Services GovCloud でのクラスターの作成** の手順を完了することで、この認証情報を使用するクラスターを作成できます。

コンソールで認証情報を編集できます。このプロバイダー接続を使用してクラスターが作成された場合には、**<cluster-namespace>** からの **<cluster-name>-aws-creds>** シークレットが新規の認証情報に更新されます。

注記: クラスタープールが要求したクラスターでは、認証情報は更新されません。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.1.2.1. S3 シークレットの作成

Amazon Simple Storage Service (S3) シークレットを作成するには、コンソールから次のタスクを実行します。

1. **Add credential** > **AWS** > **S3 Bucket** をクリックします。**For Hosted Control Plane** の場合をクリックすると、名前とネームスペースが提供されます。
2. 表示される次のフィールドに情報を入力します。
   - **bucket name**: S3 バケットの名前を追加します。

- **aws_access_key_id**: AWS アカウントの AWS アクセスキー ID を追加します。AWS にログインして ID を見つけます。
- **aws_secret_access_key**: 新しい AWS シークレットアクセスキーの内容を指定します。
- **Region**: AWS リージョンを入力します。

1.4.1.3. API を使用した不透明なシークレットの作成

API を使用して Amazon Web Services の不透明なシークレットを作成するには、次の例のような YAML プレビューウィンドウで YAML コンテンツを適用します。

```
kind: Secret
metadata:
  name: <managed-cluster-name>-aws-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  aws_access_key_id: $(echo -n "${AWS_KEY}" | base64 -w0)
  aws_secret_access_key: $(echo -n "${AWS_SECRET}" | base64 -w0)
```

注記:

- 不透明なシークレットはコンソールに表示されません。
- 不透明なシークレットは、選択したマネージドクラスターの namespace に作成されます。Hive は不透明なシークレットを使用してクラスターをプロビジョニングします。Red Hat Advanced Cluster Management コンソールを使用してクラスターをプロビジョニングすると、事前に作成された認証情報は、不透明なシークレットとしてマネージドクラスターの namespace にコピーされます。
- ラベルを認証情報に追加して、コンソールでシークレットを表示します。たとえば、以下の AWS S3 Bucket **oc label secret** に **type=awss3** および **credentials --from-file=....** が追加されます。

```
oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster "cluster.open-cluster-management.io/type=awss3"
oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster "cluster.open-cluster-management.io/credentials=credentials="
```

1.4.1.4. 関連情報

- Understanding and getting your security credentials を参照してください。
- AWS アカウントの設定 を参照してください。
- AWS にログインします。
- Red Hat OpenShift プルシークレットをダウンロードします。
- キー生成の方法は、SSH プライベートキーの生成およびエージェントへの追加 を参照してください。
- Amazon Web Services でのクラスターの作成 を参照してください。

- Amazon Web Services GovCloud でのクラスターの作成 を参照してください。
- Amazon Web Services の認証情報の作成 に戻ります。

1.4.2. Microsoft Azure の認証情報の作成

multicluster engine Operator コンソールを使用して、Microsoft Azure または Microsoft Azure Government で Red Hat OpenShift Container Platform クラスターを作成および管理するには、認証情報が必要です。

必要なアクセス権限: 編集

注記: この手順は、multicluster engine operator を使用してクラスターを作成するための前提条件です。

1.4.2.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- デプロイされた multicluster engine Operator ハブクラスター。
- Azure で Kubernetes クラスターを作成できるようにする multicluster engine Operator ハブクラスターのインターネットアクセス。
- ベースドメインのリソースグループおよび Azure Service Principal JSON などの Azure ログイン認証情報。ログイン認証情報を取得するには、**Microsoft Azure ポータル** を参照してください。
- Azre でクラスターがインストールできるようにするアカウントの権限。詳細は、**How to configure Cloud Services** および **Azure アカウントの設定** を参照してください。

1.4.2.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

1. **オプション:** 認証情報の **ベース DNS ドメイン** を追加します。ベース DNS ドメインを認証情報に追加した場合は、この認証情報でクラスターを作成すると、このベース DNS ドメインは自動的に正しいフィールドに設定されます。
2. クラスターの環境が **AzurePublicCloud** または、**AzureUSGovernmentCloud** であるかを選択します。この設定は Azure Government 環境とは異なるため、これが正しく設定されていることを確認します。
3. Azure アカウントの **ベースドメインリソースグループ名** を追加します。このエントリーは、Azure アカウントで作成したリソース名です。Azure インターフェイスで **Home** > **DNS Zones** を選択することで、ベースドメインのリソースグループ名を検索できます。ベースドメインリソースグループ名を見つけるには、**Create an Azure service principal with the Azure CLI**を参照してください。
4. **クライアント ID** の内容を入力します。この値は、以下のコマンドを使用してサービスプリンシパルを作成すると、**appId** プロパティーとして設定されます。

```
az ad sp create-for-rbac --role Contributor --name <service_principal> --scopes
<subscription_path>
```

service_principal は、お使いのサービスプリンシパル名に置き換えます。

5. **Client Secret** を追加します。この値は、以下のコマンドを使用してサービスプリンシパルを作成すると、**password** プロパティーとして設定されます。

```
az ad sp create-for-rbac --role Contributor --name <service_principal> --scopes
<subscription_path>
```

service_principal は、お使いのサービスプリンシパル名に置き換えます。

6. **Subscription ID** を追加します。以下のコマンドの出力では、この値は、**id** プロパティーになります。

```
az account show
```

7. **Tenant ID** を追加します。以下のコマンドの出力では、この値は、**tenantId** プロパティーになります。

```
az account show
```

8. プロキシーを有効にする場合は、プロキシー情報を入力します。
   - HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL。
   - HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
   - プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
   - 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。
9. **Red Hat OpenShift pull secret** を入力します。プルシークレットをダウンロードするには、**Download your Red Hat OpenShift pull secret** を参照してください。
10. クラスターへの接続に使用する **SSH 秘密鍵** と **SSH 公開鍵** を追加します。既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。

Microsoft Azure でのクラスターの作成 の手順を実行して、この認証情報を使用するクラスターを作成します。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.2.3. API を使用した不透明なシークレットの作成

コンソールの代わりに API を使用して Microsoft Azure の不透明なシークレットを作成するには、次の例のような YAML プレビューウィンドウで YAML コンテンツを適用します。

```
kind: Secret
metadata:
  name: <managed-cluster-name>-azure-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  baseDomainResourceGroupName: $(echo -n "${azure_resource_group_name}" | base64 -w0)
  osServicePrincipal.json: $(base64 -w0 "${AZURE_CRED_JSON}")
```

注記:

- 不透明なシークレットはコンソールに表示されません。
- 不透明なシークレットは、選択したマネージドクラスターの namespace に作成されます。Hive は不透明なシークレットを使用してクラスターをプロビジョニングします。Red Hat Advanced Cluster Management コンソールを使用してクラスターをプロビジョニングすると、事前に作成された認証情報は、不透明なシークレットとしてマネージドクラスターの namespace にコピーされます。

1.4.2.4. 関連情報

- Microsoft Azure Portal を参照してください。
- クラウドサービスの設定方法 を参照してください。
- Azure アカウントの設定 を参照してください。
- ベースドメインリソースグループ名を見つけるには、Create an Azure service principal with the Azure CLI を参照してください。
- Red Hat OpenShift プルシークレットをダウンロードします。
- キー生成の方法は、SSH プライベートキーの生成およびエージェントへの追加 を参照してください。
- Microsoft Azure でのクラスターの作成 を参照してください。
- Microsoft Azure の認証情報の作成 に戻ります。

1.4.3. Google Cloud Platform の認証情報の作成

multicluster engine Operator コンソールを使用して、Google Cloud Platform (GCP) で Red Hat OpenShift Container Platform クラスターを作成および管理するには、認証情報が必要です。

必要なアクセス権限: 編集

注記: この手順は、multicluster engine operator を使用してクラスターを作成するための前提条件です。

1.4.3.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- デプロイされた multicluster engine Operator ハブクラスター
- GCP で Kubernetes クラスターを作成できるようにする multicluster engine Operator ハブクラスターのインターネットアクセス
- ユーザーの Google Cloud Platform プロジェクト ID および Google Cloud Platform サービスアカウント JSON キーなど、GCP ログインの認証情報。**Creating and managing projects** を参照してください。
- GCP でクラスターがインストールできるようにするアカウントの権限。アカウントの設定方法は、**GCP プロジェクトの設定** を参照してください。

1.4.3.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

オプションで、認証情報の **ベース DNS ドメイン** を追加できます。ベース DNS ドメインを認証情報に追加した場合は、この認証情報でクラスターを作成すると、このベース DNS ドメインは自動的に正しいフィールドに設定されます。以下の手順を参照してください。

1. GCP アカウントの **Google Cloud Platform project ID** を追加します。設定を取得するには、**Log in to GCP** を参照してください。
2. **Google Cloud Platform service account JSON key** を追加します。サービスアカウントの JSON キーを作成するには、**サービスアカウントの作成** に関するドキュメントを参照してください。GCP コンソールの手順に従います。
3. 新しい **Google Cloud Platform サービスアカウントの JSON キー** の内容を提供します。
4. プロキシーを有効にする場合は、プロキシー情報を入力します。
   - HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL。
   - HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
   - プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
   - 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。
5. Red Hat OpenShift プルシークレットを入力します。プルシークレットをダウンロードするには、**Download your Red Hat OpenShift pull secret** を参照してください。
6. クラスターにアクセスできるように **SSH 秘密鍵** と **SSH 公開鍵** を追加します。既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。

Google Cloud Platform でのクラスターの作成 の手順を実行することで、クラスターの作成時にこの接続を使用できます。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.3.3. API を使用した不透明なシークレットの作成

コンソールの代わりに API を使用して Google Cloud Platform の不透明なシークレットを作成するには、次の例のような YAML プレビューウィンドウで YAML コンテンツを適用します。

```
kind: Secret
metadata:
  name: <managed-cluster-name>-gcp-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  osServiceAccount.json: $(base64 -w0 "${GCP_CRED_JSON}")
```

注記:

- 不透明なシークレットはコンソールに表示されません。
- 不透明なシークレットは、選択したマネージドクラスターの namespace に作成されます。Hive は不透明なシークレットを使用してクラスターをプロビジョニングします。Red Hat Advanced Cluster Management コンソールを使用してクラスターをプロビジョニングすると、事前に作成された認証情報は、不透明なシークレットとしてマネージドクラスターの namespace にコピーされます。

1.4.3.4. 関連情報

- Creating and managing projects を参照してください。
- GCP プロジェクトの設定 を参照してください。
- Log in to GCP .
- サービスアカウントの JSON キーを作成するには、サービスアカウントの作成 を参照してください。
- Red Hat OpenShift プルシークレットをダウンロードします。
- キー生成の方法は、SSH プライベートキーの生成およびエージェントへの追加 を参照してください。
- Google Cloud Platform でのクラスターの作成 を参照してください。

Google Cloud Platform の認証情報の作成 に戻ります。

1.4.4. VMware vSphere の認証情報の作成

multicluster engine Operator コンソールを使用して、VMware vSphere で Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。

必要なアクセス権限: 編集

注記:

- マルチクラスターエンジンオペレータを使用してクラスターを作成する前に、VMware vSphere の認証情報を作成する必要があります。
- OpenShift Container Platform バージョン 4.13 以降がサポートされます。

1.4.4.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- OpenShift Container Platform バージョン 4.13 以降にデプロイされたハブクラスター。
- VMware vSphere に Kubernetes クラスターを作成できるようにするハブクラスターのインターネットアクセス。
- インストーラーでプロビジョニングされるインフラストラクチャーを使用する場合に OpenShift Container Platform 向けに設定された VMware vSphere ログイン認証情報および vCenter 要件。**カスタマイズを使用した vSphere へのクラスターのインストール** を参照してください。これらの認証除法には、以下の情報が含まれます。
  - vCenter アカウントの権限
  - クラスターリソース
  - HDCP が利用できる
  - 時間を同期した ESXi ホスト (例: NTP)

1.4.4.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

オプションで、認証情報の **ベース DNS ドメイン** を追加できます。ベース DNS ドメインを認証情報に追加した場合は、この認証情報でクラスターを作成すると、このベース DNS ドメインは自動的に正しいフィールドに設定されます。以下の手順を参照してください。

1. **VMware vCenter サーバーの完全修飾ホスト名または IP アドレス** を追加します。値は vCenter サーバーのルート CA 証明書に定義する必要があります。可能な場合は、完全修飾ホスト名を使用します。
2. **VMware vCenter のユーザー名** を追加します。
3. **VMware vCenter パスワード** を追加します。
4. **VMware vCenter ルート CA 証明書** を追加します。
   a. VMware vCenter サーバー (**https://<vCenter_address>/certs/download.zip**) から **download.zip** として証明書をダウンロードできます。**vCenter_address** は、vCenter サーバーのアドレスに置き換えます。
   b. **download.zip** のパッケージを展開します。

c. 拡張子が **.0** の **certs/<platform>** ディレクトリーの証明書を使用します。
ヒント: ls certs/<platform> コマンドを使用して、お使いのプラットフォームで使用可能な全証明書を一覧表示できます。

<platform> は、**lin**、**mac**、または **win** など、お使いのプラットフォームに置き換えます。

例: **certs/lin/3a343545.0**

ベストプラクティス: cat certs/lin/*.0 > ca.crt コマンドを実行して、拡張子 **.0** を持つ複数の証明書をリンクします。

d. **VMware vSphere クラスター名** を追加します。

e. **VMware vSphere データセンター** を追加します。

f. **VMware vSphere デフォルトデータストア** を追加します。

g. **VMware vSphere ディスクタイプ** を追加します。

h. **VMware vSphere フォルダー** を追加します。

i. **VMware vSphere リソースプール** を追加します。

5. オフラインインストールのみ: **Configuration for disconnected installation**サブセクションのフィールドに必要な情報を入力します。

- **Cluster OS image**: この値には、Red Hat OpenShift Container Platform クラスターマシンに使用するイメージの URL が含まれます。

- **Image content source**: この値には、オフラインのレジストリーパスが含まれます。このパスには、オフラインインストールに使用する全インストールイメージのホスト名、ポート、レジストリーパスが含まれます。たとえば、**repository.com:5000/openshift/ocp-release** となります。
このパスは、Red Hat OpenShift Container Platform リリースイメージに対して、**install-config.yaml** のイメージコンテンツソースポリシーのマッピングを作成します。たとえば、**repository.com:5000** は以下の **imageContentSource** コンテンツを作成します。

```
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-release-nightly
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-release
- mirrors:
  - registry.example.com:5000/ocp4
  source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
```

- **Additional trust bundle**: この値で、ミラーレジストリーへのアクセスに必要な証明書ファイルのコンテンツを指定します。
注記: 非接続環境にあるハブクラスターからマネージドクラスターをデプロイして、インストール後の設定を自動的にインポートする場合は、**YAML** エディターを使用してイメージコンテンツソースポリシーを **install-config.yaml** ファイルに追加します。エントリーの例を以下に示します。

```
- mirrors:
  - registry.example.com:5000/rhacm2
  source: registry.redhat.io/rhacm2
```

6. プロキシーを有効にする場合は、プロキシー情報を入力します。
   - HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL。
   - HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
   - プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
   - 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。
7. Red Hat OpenShift プルシークレットを入力します。プルシークレットをダウンロードするには、**Download your Red Hat OpenShift pull secret**を参照してください。
8. **SSH 秘密鍵** と **SSH 公開鍵** を追加し、クラスターに接続できるようにします。
   既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。

VMware vSphere でのクラスターの作成 の手順を完了することで、この認証情報を使用するクラスターを作成できます。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.4.3. API を使用した不透明なシークレットの作成

コンソールの代わりに API を使用して VMware vSphere の不透明なシークレットを作成するには、次の例のような YAML プレビューウィンドウで YAML コンテンツを適用します。

```
kind: Secret
metadata:
   name: <managed-cluster-name>-vsphere-creds
   namespace: <managed-cluster-namespace>
type: Opaque
data:
   username: $(echo -n "${VMW_USERNAME}" | base64 -w0)
   password.json: $(base64 -w0 "${VMW_PASSWORD}")
```

注記:

- 不透明なシークレットはコンソールに表示されません。
- 不透明なシークレットは、選択したマネージドクラスターの namespace に作成されます。Hive は不透明なシークレットを使用してクラスターをプロビジョニングします。Red Hat Advanced Cluster Management コンソールを使用してクラスターをプロビジョニングすると、事前に作成された認証情報は、不透明なシークレットとしてマネージドクラスターの namespace にコピーされます。

1.4.4.4. 関連情報

- カスタマイズを使用した vSphere へのクラスターのインストール を参照してください。
- Red Hat OpenShift プルシークレットをダウンロードします。
- 詳細は、クラスターノード SSH アクセス用のキーペアの生成 を参照してください。
- VMware vSphere でのクラスターの作成 を参照してください。
- VMware vSphere の認証情報の作成 に戻ります。

1.4.5. Red Hat OpenStack の認証情報の作成

multicluster engine Operator コンソールを使用して、Red Hat OpenStack Platform で Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。

注記:

- Multi-Cluster Engine Operator を使用してクラスターを作成する前に、Red Hat OpenStack Platform の認証情報を作成する必要があります。
- OpenShift Container Platform バージョン 4.13 以降のみがサポートされます。

1.4.5.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- OpenShift Container Platform バージョン 4.13 以降にデプロイされたハブクラスター。
- Red Hat OpenStack Platform で Kubernetes クラスターを作成できるようにするハブクラスターのインターネットアクセス。
- インストーラーでプロビジョニングされるインフラストラクチャーを使用する場合に OpenShift Container Platform 向けに設定された Red Hat OpenStack Platform ログイン認証情報および Red Hat OpenStack Platform の要件。**カスタマイズを使用した OpenStack へのクラスターのインストール** を参照してください。
- CloudStack API にアクセスするための **clouds.yaml** ファイルをダウンロードまたは作成する。**clouds.yaml** ファイルで以下を行います。
  - 使用する cloud auth セクション名を決定します。
  - **username** 行の直後に、**password** の行を追加します。

1.4.5.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。セキュリティーと利便性を強化するために、認証情報をホストするためだけに namespace を作成できます。

1. **オプション:** 認証情報のベース DNS ドメインを追加できます。ベース DNS ドメインを追加すると、この認証情報を使用してクラスターを作成するときに、正しいフィールドに自動的に入力されます。

2. Red Hat OpenStack Platform の **clouds.yaml** ファイルの内容を追加します。パスワードを含む **clouds.yaml** ファイルの内容で、Red Hat OpenStack Platform サーバーへの接続に必要な情報を提供します。ファイルの内容には、**username** の直後に新たに追加したパスワードを含める必要があります。

3. Red Hat OpenStack Platform クラウド名を追加します。このエントリーは、Red Hat OpenStack Platform サーバーへの通信確立に使用する **clouds.yaml** の cloud セクションで指定した名前です。

4. **オプション**: 内部認証局を使用する設定の場合は、**内部 CA 証明書** フィールドに証明書を入力して、証明書情報で **Clouds.yaml** を自動的に更新します。

5. オフラインインストールのみ: **Configuration for disconnected installation**サブセクションのフィールドに必要な情報を入力します。

   - **Cluster OS image**: この値には、Red Hat OpenShift Container Platform クラスターマシンに使用するイメージの URL が含まれます。

   - **イメージコンテンツソース**: この値には、オフラインのレジストリーパスが含まれます。このパスには、オフラインインストールに使用する全インストールイメージのホスト名、ポート、レジストリーパスが含まれます。たとえば、**repository.com:5000/openshift/ocp-release** となります。
     このパスは、Red Hat OpenShift Container Platform リリースイメージに対して、**install-config.yaml** のイメージコンテンツソースポリシーのマッピングを作成します。たとえば、**repository.com:5000** は以下の **imageContentSource** コンテンツを作成します。

     ```
     - mirrors:
       - registry.example.com:5000/ocp4
       source: quay.io/openshift-release-dev/ocp-release-nightly
     - mirrors:
       - registry.example.com:5000/ocp4
       source: quay.io/openshift-release-dev/ocp-release
     - mirrors:
       - registry.example.com:5000/ocp4
       source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
     ```

   - **Additional trust bundle**: この値で、ミラーレジストリーへのアクセスに必要な証明書ファイルのコンテンツを指定します。
     注記: 非接続環境にあるハブクラスターからマネージドクラスターをデプロイして、インストール後の設定を自動的にインポートする場合は、**YAML** エディターを使用してイメージコンテンツソースポリシーを **install-config.yaml** ファイルに追加します。エントリーの例を以下に示します。

     ```
     - mirrors:
       - registry.example.com:5000/rhacm2
       source: registry.redhat.io/rhacm2
     ```

6. プロキシーを有効にする必要がある場合は、プロキシー情報を入力します。

   - HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL。

   - HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。

- プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。

- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

7. Red Hat OpenShift プルシークレットを入力します。プルシークレットをダウンロードするには、**Download your Red Hat OpenShift pull secret**を参照してください。

8. SSH 秘密鍵と SSH 公開鍵を追加し、クラスターに接続できるようにします。既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。

9. **Create** をクリックします。

10. 新規の認証情報を確認し、**Add** をクリックします。認証情報を追加すると、認証情報のリストに追加されます。

Red Hat OpenStack Platform でのクラスターの作成 の手順を実行して、この認証情報を使用するクラスターを作成します。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.5.3. API を使用した不透明なシークレットの作成

コンソールの代わりに API を使用して Red Hat OpenStack Platform の不透明なシークレットを作成するには、次の例のような YAML プレビューウィンドウで YAML コンテンツを適用します。

```
kind: Secret
metadata:
  name: <managed-cluster-name>-osp-creds
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  clouds.yaml: $(base64 -w0 "${OSP_CRED_YAML}") cloud: $(echo -n "openstack" | base64 -w0)
```

注記:

- 不透明なシークレットはコンソールに表示されません。

- 不透明なシークレットは、選択したマネージドクラスターの namespace に作成されます。Hive は不透明なシークレットを使用してクラスターをプロビジョニングします。Red Hat Advanced Cluster Management コンソールを使用してクラスターをプロビジョニングすると、事前に作成された認証情報は、不透明なシークレットとしてマネージドクラスターの namespace にコピーされます。

1.4.5.4. 関連情報

- カスタマイズを使用した OpenStack へのクラスターのインストール を参照してください。

- Red Hat OpenShift プルシークレットをダウンロードします。

- 詳細は、クラスターノード SSH アクセス用のキーペアの生成 を参照してください。

- Red Hat OpenStack Platform でのクラスターの作成 を参照してください。
- Red Hat OpenStack の認証情報の作成 に戻ります。

1.4.6. Red Hat Virtualization の認証情報の作成

非推奨: Red Hat Virtualization 認証情報とクラスター作成機能は非推奨となり、サポートされなくなりました。

multicluster engine Operator コンソールを使用して、Red Hat Virtualization で Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。

注記: この手順は、multicluster engine operator でクラスターを作成する前に実行する必要があります。

1.4.6.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- OpenShift Container Platform バージョン 4.13 以降にデプロイされたハブクラスター。
- Red Hat Virtualization で Kubernetes クラスターを作成できるようにするハブクラスターのインターネットアクセス。
- 設定済み Red Hat Virtualization 環境の Red Hat Virtualization ログイン認証情報。Red Hat Virtualization ドキュメントの インストールガイド を参照してください。以下のリストは、必要な情報を示しています。
  - oVirt URL
  - ovirt 完全修飾ドメイン名 (FQDN)
  - oVirt ユーザー名
  - oVirt パスワード
  - oVirt CA/証明書
- オプション: プロキシーを有効にした場合にはプロキシー情報。
- Red Hat OpenShift Container Platform のプルシークレット情報。Pull secret からプルシークレットをダウンロードします。
- 最終的なクラスターの情報を転送するための SSH 秘密鍵と公開鍵。
- oVirt でクラスターをインストールできるようにするアカウントの権限。

1.4.6.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー向上のため、認証情報のホスト専用の namespace を作成します。

1. 新しい認証情報の基本情報を追加します。オプションで、この認証情報を使用してクラスターを作成すると自動的に正しいフィールドにデータが投入される Base DNS ドメインを追加できます。認証情報に追加しない場合は、クラスターの作成時に追加できます。
2. Red Hat Virtualization 環境に必要な情報を追加します。
3. プロキシーを有効にする必要がある場合は、プロキシー情報を入力します。
   - HTTP Proxy URL: **HTTP** トラフィックのプロキシーとして使用する URL。
   - HTTPS Proxy URL: **HTTPS** トラフィックに使用する必要のあるセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
   - プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
4. Red Hat OpenShift Container Platform プルシークレットを入力します。Pull secret からプルシークレットをダウンロードします。
5. SSH 秘密鍵と SSH 公開鍵を追加し、クラスターに接続できるようにします。既存のキーペアを使用するか、キー生成プログラムで新しいキーを作成できます。詳細は、クラスターノード SSH アクセス用のキーペアの生成 を参照してください。
6. 新規の認証情報を確認し、**Add** をクリックします。認証情報を追加すると、認証情報のリストに追加されます。

Red Hat Virtualization でのクラスターの作成 (非推奨) の手順を実行して、この認証情報を使用するクラスターを作成します。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.7. Red Hat OpenShift Cluster Manager の認証情報の作成

クラスターを検出できるように OpenShift Cluster Manager の認証情報を追加します。

必要なアクセス権限: 管理者

1.4.7.1. 前提条件

cloud.redhat.com アカウントへのアクセスが必要です。console.redhat.com/openshift/token から取得できる値が後で必要になります。

1.4.7.2. コンソールを使用した認証情報の管理

クラスター検出用の認証情報を追加する必要があります。multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

OpenShift Cluster Manager API トークンは、 console.redhat.com/openshift/token から取得できます。

コンソールで認証情報を編集できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

認証情報が削除されるか、OpenShift Cluster Manager API トークンの有効期限が切れるか、取り消されると、関連付けられた検出クラスターが削除されます。

1.4.8. Ansible Automation Platform の認証情報の作成

multicluster engine Operator コンソールを使用して、Red Hat Ansible Automation Platform を使用する Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。

必要なアクセス権限: 編集

注記: この手順は、自動化テンプレートを作成して、クラスターで自動化を有効にする前に、実行する必要があります。

1.4.8.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- デプロイされた multicluster engine Operator ハブクラスター
- multicluster engine Operator ハブクラスターのインターネットアクセス
- Ansible Automation Platform ホスト名と OAuth トークンを含む Ansible ログイン認証情報。Ansible Automation Platform の認証情報 を参照してください。
- ハブクラスターのインストールおよび Ansible 操作をできるようにするアカウント権限。Ansible ユーザー の詳細を確認してください。

1.4.8.2. コンソールを使用した認証情報の管理

multicluster engine Operator コンソールから認証情報を作成するには、コンソールで手順を実行します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

Ansible 認証情報の作成時に指定する Ansible トークンとホストの URL は、認証情報の編集時にその認証情報を使用する自動化向けに、自動で更新されます。更新は、クラスターライフサイクル、ガバナンス、およびアプリケーション管理の自動化に関連するものなど、Ansible 認証情報を使用する自動化にコピーされます。これにより、認証情報の更新後も自動化が引き続き実行されます。

コンソールで認証情報を編集できます。Ansible 認証情報は、認証情報の更新時に、対象の認証情報を使用する自動化で、自動的に更新されあす。

マネージドクラスターで実行する Ansible Automation Platform タスクの設定 の手順を完了することで、この認証情報を使用する Ansible ジョブを作成できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.4.9. オンプレミス環境の認証情報の作成

コンソールを使用してオンプレミス環境で Red Hat OpenShift Container Platform クラスターをデプロイおよび管理するには、認証情報が必要です。認証情報では、クラスターに使用される接続を指定します。

必要なアクセス権限: 編集

- 前提条件
- コンソールを使用した認証情報の管理

1.4.9.1. 前提条件

認証情報を作成する前に、以下の前提条件を満たす必要があります。

- デプロイされたハブクラスター。
- インフラストラクチャー環境に Kubernetes クラスターを作成できるようにするハブクラスターのインターネットアクセス。
- オフライン環境では、クラスター作成用のリリースイメージをコピーできるミラーレジストリーを設定している。詳細は、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 を参照してください。
- オンプレミス環境でのクラスターのインストールをサポートするアカウントの権限。

1.4.9.2. コンソールを使用した認証情報の管理

コンソールから認証情報を作成するには、コンソールで手順を完了します。

ナビゲーションメニューから開始します。**Credentials** をクリックし、既存の認証情報オプションから選択します。**ヒント:** 便宜上およびセキュリティー上、認証情報のホスト専用の namespace を作成します。

1. 認証情報の種類に **Host inventory** を選択します。
2. オプションで、認証情報の **ベース DNS ドメイン** を追加できます。ベース DNS ドメインを認証情報に追加した場合は、この認証情報でクラスターを作成すると、このベース DNS ドメインは自動的に正しいフィールドに設定されます。DNS ドメインを追加していない場合は、クラスターの作成時に追加できます。
3. **Red Hat OpenShift pull secret** を入力します。このプルシークレットは、クラスターを作成してこの認証情報を指定すると、自動的に入力されます。Pull secret からプルシークレットをダウンロードします。プルシークレットの詳細は、イメージプルシークレットの使用 を参照してください。
4. **SSH public key** を入力します。この **SSH public key** クラスターを作成してこの認証情報を指定するときにも自動的に入力されます。
5. **Add** を選択して認証情報を作成します。

オンプレミス環境でのクラスターの作成 の手順を完了することで、この認証情報を使用するクラスターを作成できます。

認証情報を使用するクラスターの管理を終了する場合は、認証情報を削除して認証情報内にある情報を保護します。**Actions** を選択して、一括削除するか、削除する認証情報の横にあるオプションメニューを選択します。

1.5. クラスターライフサイクルの概要

multicluster engine Operator は、OpenShift Container Platform および Red Hat Advanced Cluster Management ハブクラスターにクラスター管理機能を提供するクラスターライフサイクル Operator です。multicluster engine Operator は、クラスターフリート管理を強化し、クラウドおよびデータセンター全体の OpenShift Container Platform クラスターライフサイクル管理をサポートするソフトウェア Operator です。Red Hat Advanced Cluster Management の有無にかかわらず、multicluster engine Operator を使用できます。Red Hat Advanced Cluster Management は、multicluster engine Operator を自動的にインストールし、さらにマルチクラスター機能を提供します。

以下のドキュメントを参照してください。

- クラスターライフサイクルのアーキテクチャー
- 認証情報の管理の概要
- リリースイメージ
  - リリースイメージの指定
  - 接続中にリリースイメージのカスタムリストを維持する
  - 非接続時におけるリリースイメージのカスタム一覧の管理
- ホストインベントリーの概要
- クラスター作成
  - CLI を使用したクラスターの作成
  - クラスター作成時の追加のマニフェストの設定
  - Amazon Web Services でのクラスターの作成
  - Amazon Web Services GovCloud でのクラスターの作成
  - Microsoft Azure でのクラスターの作成
  - Google Cloud Platform でのクラスターの作成
  - VMware vSphere でのクラスターの作成
  - Red Hat OpenStack Platform でのクラスターの作成
  - Red Hat Virtualization でのクラスターの作成 (非推奨)
  - オンプレミス環境でのクラスターの作成
  - プロキシー環境でのクラスターの作成
- クラスターのインポート

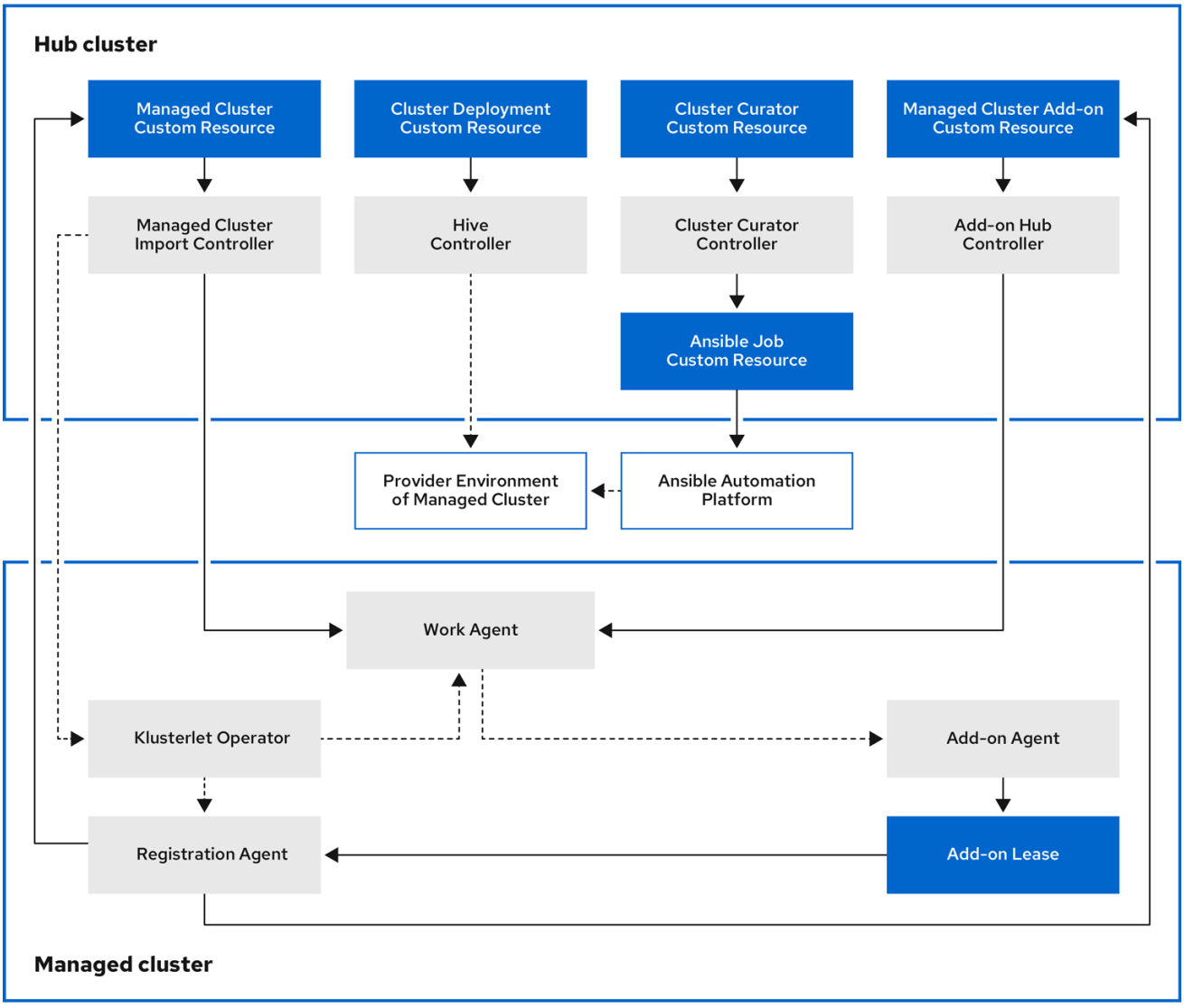
1.5.1. クラスターライフサイクルのアーキテクチャー

クラスターライフサイクルには、**ハブクラスター** と **マネージドクラスター** の 2 種類のクラスターが必要です。

ハブクラスターは、multicluster engine Operator が自動的にインストールされる OpenShift Container Platform (または Red Hat Advanced Cluster Management) メインクラスターです。ハブクラスターを使用して他の Kubernetes クラスターの作成、管理、および監視を行うことができます。ハブクラスターを使用してクラスターを作成できますが、ハブクラスターが管理する既存のクラスターをインポートすることもできます。

マネージドクラスターを作成すると、クラスターは Red Hat OpenShift Container Platform クラスターインストーラーと Hive リソースを使用して作成されます。OpenShift Container Platform インストーラーでクラスターをインストールするプロセスについての詳細は、OpenShift Container Platform ドキュメントの OpenShift Container Platform インストールの概要 を参照してください。

次の図は、クラスター管理に使用する Kubernetes Operator 用のマルチクラスターエンジンと共にインストールされるコンポーネントを示しています。



303_RHACM_0223

クラスターライフサイクル管理のアーキテクチャーのコンポーネントには、以下の項目が含まれます。

1.5.1.1. ハブクラスター

- **マネージドクラスターのインポートコントローラー** は、klusterlet Operator をマネージドクラスターにデプロイします。
- **Hive コントローラー** は、Kubernetes Operator 用のマルチクラスターエンジンを使用して作成したクラスターをプロビジョニングします。また、Hive コントローラーは、Kubernetes Operator 用のマルチクラスターエンジンによって作成されたマネージドクラスターを破棄します。
- **クラスターキュレーターコントローラー** は、マネージドクラスターの作成またはアップグレード時にクラスターインフラストラクチャー環境を設定するためのプレフックまたはポストフックとして Ansible ジョブを作成します。
- マネージドクラスターアドオンがハブクラスターで有効になると、その **アドオンハブコントローラー** がハブクラスターにデプロイされます。**アドオンハブコントローラー** は、**アドオンエージェント** をマネージドクラスターにデプロイします。

1.5.1.2. マネージドクラスター

- **klusterlet Operator** マネージドクラスターに登録およびワークコントローラーをデプロイします。
- **登録エージェント** は、マネージドクラスターとマネージドクラスターアドオンをハブクラスターに登録します。登録エージェントは、管理対象クラスターと管理対象クラスターアドオンのステータスも維持します。次のアクセス許可が Clusterrole 内に自動的に作成され、マネージドクラスターがハブクラスターにアクセスできるようになります。
  - エージェントは、ハブクラスターが管理する所有クラスターを取得または更新できます。
  - エージェントが、ハブクラスターが管理する所有クラスターのステータスを更新できるようにします。
  - エージェントが証明書をローテーションできるようにします。
  - エージェントが **coordination.k8s.io** リースを **get** または **update** できるようにします。
  - エージェントがマネージドクラスターアドオンを **get** できるようにします。
  - エージェントがマネージドクラスターアドオンのステータスを更新できるようにします。
- **ワークエージェント** は、アドオンエージェントをマネージドクラスターに適用します。マネージドクラスターによるハブクラスターへのアクセスを許可する権限は、Clusterrole 内に自動的に作成され、エージェントはイベントをハブクラスターに送信できます。

クラスターの追加と管理を続行するには、クラスターライフサイクルの概要 を参照してください。

1.5.2. リリースイメージ

クラスターをビルドするときは、リリースイメージで指定されているバージョンの Red Hat OpenShift Container Platform を使用します。デフォルトでは、OpenShift Container Platform は **clusterImageSets** リソースを使用して、サポートされているリリースイメージのリストを取得します。

リリースイメージの詳細については、読み続けてください。

- リリースイメージの指定

- 接続中にリリースイメージのカスタムリストを維持する
- 非接続時におけるリリースイメージのカスタム一覧の管理

1.5.2.1. リリースイメージの指定

Kubernetes Operator 用のマルチクラスターエンジンを使用してプロバイダー上にクラスターを作成する場合は、新しいクラスターに使用するリリースイメージを指定します。リリースイメージを指定するには、次のトピックを参照してください。

- ClusterImageSets の検索
- ClusterImageSets の設定
- 別のアーキテクチャーにクラスターをデプロイするためのリリースイメージの作成

1.5.2.1.1. ClusterImageSets の検索

リリースイメージを参照する YAML ファイルは、**acm-hive-openshift-releases** GitHub リポジトリーに保持されます。これらのファイルは、コンソールで利用可能なリリースイメージのリストを作成します。これには、OpenShift Container Platform における最新の fast チャネルイメージが含まれます。

コンソールには、OpenShift Container Platform の 3 つの最新バージョンの最新リリースイメージのみが表示されます。たとえば、コンソールオプションに以下のリリースイメージが表示される可能性があります。

quay.io/openshift-release-dev/ocp-release:4.13.1-x86_64

コンソールには最新バージョンが表示され、最新のリリースイメージを使用してクラスターを作成するのに役立ちます。特定のバージョンのクラスターを作成する必要がある場合は、古いリリースイメージバージョンも利用できます。

注記: コンソールでクラスターを作成する場合は、**visible: 'true'** ラベルを持つイメージのみを選択できます。**ClusterImageSet** リソース内のこのラベルの例は、以下の内容で提供されます。**4.x.1** は、製品の最新バージョンに置き換えます。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.x.1-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.1-x86_64
```

追加のリリースイメージは保管されますが、コンソールには表示されません。利用可能なリリースイメージをすべて表示するには、次のコマンドを実行します。

```
oc get clusterimageset
```

リポジトリーには、**clusterImageSets** ディレクトリーがあります。これは、リリースイメージを操作するときに使用するディレクトリーです。**clusterImageSets** ディレクトリーには、次のディレクトリーがあります。

- Fast: サポート対象の各 OpenShift Container Platform バージョンのリリースイメージの内、最新バージョンを参照するファイルが含まれます。このフォルダー内のリリースイメージはテストされ、検証されており、サポートされます。

- Releases: 各 OpenShift Container Platform バージョン (stable、fast、および candidate チャネル) のリリースイメージすべてを参照するファイルが含まれます。
  注記: これらのリリースすべてがテストされおらず、安定版とみなされているわけではありません。

- Stable: サポート対象の各 OpenShift Container Platform バージョンのリリースイメージの内、最新の安定版 2 つを参照するファイルが含まれます。
  注記: デフォルトでは、リリースイメージの現在のリストは 1 時間ごとに更新されます。製品をアップグレードした後、リストに製品の新しいバージョンの推奨リリースイメージバージョンが反映されるまでに最大 1 時間かかる場合があります。

1.5.2.1.2. ClusterImageSets の設定

次のオプションを使用して **ClusterImageSets** を設定できます。

- オプション 1: コンソールでクラスターを作成するには、使用する特定の **ClusterImageSet** のイメージ参照を指定します。指定した新しいエントリーはそれぞれ保持され、将来のすべてのクラスタープロビジョニングで使用できます。次のエントリーの例を参照してください。

  ```
  quay.io/openshift-release-dev/ocp-release:4.6.8-x86_64
  ```

- オプション 2: GitHub リポジトリー **acm-hive-openshift-releases** から YAML ファイル **ClusterImageSets** を手動で作成し、適用します。

- オプション 3: フォークされた GitHub リポジトリーから **ClusterImageSets** の自動更新を有効にするには、**cluster-image-set-controller** GitHub リポジトリーの **README.md** に従います。

1.5.2.1.3. 別のアーキテクチャーにクラスターをデプロイするためのリリースイメージの作成

両方のアーキテクチャーのファイルを持つリリースイメージを手動で作成することで、ハブクラスターのアーキテクチャーとは異なるアーキテクチャーでクラスターを作成できます。

たとえば、**ppc64le**、**aarch64**、または **s390x** アーキテクチャーで実行されているハブクラスターから **x86_64** クラスターを作成する必要があるとします。両方のファイルセットでリリースイメージを作成する場合に、新規のリリースイメージにより OpenShift Container Platform リリースレジストリーがマルチアーキテクチャーイメージマニフェストを提供できるので、クラスターの作成は成功します。

OpenShift Container Platform 4.13 以降では、デフォルトで複数のアーキテクチャーがサポートされます。以下の **clusterImageSet** を使用してクラスターをプロビジョニングできます。**4.x.0** は、最新バージョンに置き換えます。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.x.0-multi-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.0-multi
```

複数のアーキテクチャーをサポートしない OpenShift Container Platform イメージのリリースイメージを作成するには、アーキテクチャータイプについて以下のような手順を実行します。

1. OpenShift Container Platform リリースレジストリー から、**x86_64**、**s390x**、**aarch64**、および **ppc64le** リリースイメージを含む マニフェスト一覧 を作成します。

   a. 以下のコマンド例を実行して、Quay リポジトリー から環境内の両方のアーキテクチャーのマニフェストリストをプルします。**4.x.1** は、製品の最新バージョンに置き換えます。

   ```
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-x86_64
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-ppc64le
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-s390x
   podman pull quay.io/openshift-release-dev/ocp-release:4.x.1-aarch64
   ```

   b. 次のコマンドを実行して、イメージを管理するプライベートリポジトリーにログインします。**<private-repo>** は、リポジトリーへのパスに置き換えます。

   ```
   podman login <private-repo>
   ```

   c. 環境に適用される以下のコマンドを実行して、リリースイメージマニフェストをプライベートリポジトリーに追加します。**4.x.1** は、製品の最新バージョンに置き換えます。**<private-repo>** は、リポジトリーへのパスに置き換えます。

   ```
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-x86_64 <private-repo>/ocp-release:4.x.1-x86_64
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-ppc64le <private-repo>/ocp-release:4.x.1-ppc64le
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-s390x <private-repo>/ocp-release:4.x.1-s390x
   podman push quay.io/openshift-release-dev/ocp-release:4.x.1-aarch64 <private-repo>/ocp-release:4.x.1-aarch64
   ```

   d. 次のコマンドを実行して、新しい情報のマニフェストを作成します。

   ```
   podman manifest create mymanifest
   ```

   e. 次のコマンドを実行して、両方のリリースイメージへの参照をマニフェストリストに追加します。**4.x.1** は、製品の最新バージョンに置き換えます。**<private-repo>** は、リポジトリーへのパスに置き換えます。

   ```
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-x86_64
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-ppc64le
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-s390x
   podman manifest add mymanifest <private-repo>/ocp-release:4.x.1-aarch64
   ```

   f. 次のコマンドを実行して、マニフェストリスト内のリストを既存のマニフェストとマージします。**<private-repo>** は、リポジトリーへのパスに置き換えます。**4.x.1** は、最新バージョンに置き換えます。

   ```
   podman manifest push mymanifest docker://<private-repo>/ocp-release:4.x.1
   ```

2. ハブクラスターで、リポジトリーのマニフェストを参照するリリースイメージを作成します。

   a. 以下の例のような情報を含む YAML ファイルを作成します。**<private-repo>** は、リポジト

リーへのパスに置き換えます。**4.x.1** は、最新バージョンに置き換えます。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: "true"
  name: img4.x.1-appsub
spec:
  releaseImage: <private-repo>/ocp-release:4.x.1
```

b. ハブクラスターで以下のコマンドを実行し、変更を適用します。**<file-name>** を、先の手順で作成した YAML ファイルの名前に置き換えます。

```
oc apply -f <file-name>.yaml
```

3. OpenShift Container Platform クラスターの作成時に新規リリースイメージを選択します。

4. Red Hat Advanced Cluster Management コンソールを使用してマネージドクラスターをデプロイする場合は、クラスター作成プロセス時に **Architecture** フィールドにマネージドクラスターのアーキテクチャーを指定します。

作成プロセスでは、マージされたリリースイメージを使用してクラスターを作成します。

1.5.2.1.4. 関連情報

- リリースイメージを参照する YAML ファイルについては、acm-hive-openshift-releases GitHub リポジトリーを参照してください。
- フォークされた GitHub リポジトリーから **ClusterImageSets** の自動更新を有効にする方法については、cluster-image-set-controller GitHub リポジトリーを参照してください。

1.5.2.2. 接続時におけるリリースイメージのカスタム一覧の管理

すべてのクラスターに同じリリースイメージを使用することもできます。クラスターの作成時に利用可能なリリースイメージのカスタムリストを作成し、作業を簡素化します。利用可能なリリースイメージを管理するには、以下の手順を実行します。

1. acm-hive-openshift-releases GitHub をフォークします。
2. クラスターの作成時に使用できるイメージの YAML ファイルを追加します。Git コンソールまたはターミナルを使用して、イメージを **./clusterImageSets/stable/** または **./clusterImageSets/fast/** ディレクトリーに追加します。
3. **cluster-image-set-git-repo** という名前の **multicluster-engine** namespace に **ConfigMap** を作成します。次の例を参照してください。ただし、**2.x** は 2.5 に置き換えてください。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-image-set-git-repo
  namespace: multicluster-engine
data:
  gitRepoUrl: <forked acm-hive-openshift-releases repository URL>
```

```
gitRepoBranch: backplane-<2.x>
gitRepoPath: clusterImageSets
channel: <fast or stable>
```

次の手順でフォークされたリポジトリーに変更をマージすることで、メインリポジトリーから利用可能な YAML ファイルを取得できます。

1. フォークしたリポジトリーに変更をコミットし、マージします。
2. **acm-hive-openshift-releases** リポジトリーのクローンを作成した後に高速リリースイメージのリストを同期するには、**cluster-image-set-git-repo ConfigMap** の channel フィールドの値を **fast** に更新します。
3. 安定版リリースイメージを同期して表示するには、**cluster-image-set-git-repo ConfigMap** の Channel フィールドの値を **steady** に更新します。

ConfigMap を更新すると、約 1 分以内に、利用可能な安定リリースイメージのリストが現在利用可能なイメージで更新されます。

1. 以下のコマンドを使用して、利用可能ものを表示し、デフォルトの設定を削除します。**<clusterImageSet_NAME>** を正しい名前に置き換えます。

```
oc get clusterImageSets
oc delete clusterImageSet <clusterImageSet_NAME>
```

クラスターの作成時に、コンソールで現在利用可能なリリースイメージの一覧を表示します。

ConfigMap を通じて利用できる他のフィールドについては、cluster-image-set-controller GitHub リポジトリーの README を参照してください。

1.5.2.3. 非接続時におけるリリースイメージのカスタム一覧の管理

ハブクラスターにインターネット接続がない場合は、リリースイメージのカスタムリストを管理しないといけない場合があります。クラスターの作成時に利用可能なリリースイメージのカスタムリストを作成します。非接続時に、利用可能なリリースイメージを管理するには、以下の手順を実行します。

1. オンライン接続されているシステムを使用している場合は、acm-hive-openshift-releases GitHub リポジトリー に移動し、利用可能なクラスターイメージセットにアクセスします。
2. **clusterImageSets** ディレクトリーを非接続 multicluster engine Operator クラスターにアクセスできるシステムにコピーします。
3. 管理対象クラスターに適合する次の手順を実行して、管理対象クラスターとオフラインリポジトリーの間にクラスターイメージセットを含むマッピングを追加します。
   - OpenShift Container Platform マネージドクラスターの場合は、**ImageContentSourcePolicy** オブジェクトを使用してマッピングを完了する方法についての詳細は、イメージレジストリーリポジトリーのミラーリングの設定 を参照してください。
   - OpenShift Container Platform クラスターではないマネージドクラスターの場合は、**ManageClusterImageRegistry** カスタムリソース定義を使用してイメージセットの場所を上書きします。マッピング用にクラスターを上書きする方法については、インポート用のマネージドクラスターでのレジストリーイメージの指定 を参照してください。

4. コンソールまたは CLI を使用して、**clusterImageSet** YAML コンテンツを手動で追加することにより、クラスターを作成するときに使用できるイメージの YAML ファイルを追加します。

5. 残りの OpenShift Container Platform リリースイメージの **clusterImageSet** YAML ファイルを、イメージの保存先の正しいオフラインリポジトリーを参照するように変更します。更新内容は次の例のようになります。ここでは、**spec.releaseImage** が使用しているイメージレジストリーを参照しています。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
  name: img4.13.8-x86-64-appsub
spec:
  releaseImage: IMAGE_REGISTRY_IPADDRESS_or_DNSNAME/REPO_PATH/ocp-release:4.13.8-x86_64
```

   YAML ファイルで参照されているオフラインイメージレジストリーにイメージがロードされていることを確認します。

6. 各 YAML ファイルに以下のコマンドを入力して、各 **clusterImageSets** を作成します。

```
oc create -f <clusterImageSet_FILE>
```

   clusterImageSet_FILE を、クラスターイメージセットファイルの名前に置き換えます。以下に例を示します。

```
oc create -f img4.11.9-x86_64.yaml
```

   追加するリソースごとにこのコマンドを実行すると、使用可能なリリースイメージのリストが表示されます。

7. または、クラスターの作成コンソールに直接イメージ URL を貼り付けることもできます。イメージ URL を追加すると、新しい **clusterImageSets** が存在しない場合に作成されます。

8. クラスターの作成時に、コンソールで現在利用可能なリリースイメージの一覧を表示します。

1.5.3. ホストインベントリーの概要

ホストインベントリー管理とオンプレミスクラスターのインストールは、multicluster engine Operator の central infrastructure management 機能を使用して利用できます。central infrastructure management は、Assisted Installer (infrastructure Operator とも呼ばれる) をハブクラスターの Operator として実行します。

コンソールを使用してホストインベントリーを作成できます。これは、オンプレミスの OpenShift Container Platform クラスターの作成に使用できるベアメタルまたは仮想マシンのプールです。これらのクラスターは、コントロールプレーン専用のマシンまたは Hosted control plane を備えたスタンドアロンにすることができます。ここでは、コントロールプレーンはハブクラスター上の Pod として実行されれます。

スタンドアロンクラスターは、コンソール、API、またはゼロタッチプロビジョニング (ZTP) を使用する GitOps を使用してインストールできます。詳細は、Red Hat OpenShift Container Platform ドキュメントの 非接続環境での GitOps ZTP のインストール を参照してください。

マシンは、Discovery Image で起動した後、ホストインベントリーに参加します。Discovery Image は、以下を含む Red Hat CoreOS ライブイメージです。

- 検出、検証、およびインストールタスクを実行するエージェント。
- ハブクラスター上のサービスにアクセスするために必要な設定 (該当する場合、エンドポイント、トークン、静的ネットワーク設定など)。

通常、インフラストラクチャー環境ごとに 1 つの Discovery Image があり、これは共通のプロパティーセットを共有するホストのセットです。**InfraEnv** カスタムリソース定義は、このインフラストラクチャー環境と関連する Discovery Image を表します。使用されるイメージは OpenShift Container Platform のバージョンに基づいており、選択したオペレーティングシステムのバージョンが決まります。

ホストが起動し、エージェントがサービスに接続すると、サービスはそのホストを表すハブクラスター上に新しい **Agent** カスタムリソースを作成します。**Agent** リソースはホストインベントリーを設定します。

後でホストを OpenShift ノードとしてインベントリーにインストールできます。エージェントは、必要な設定とともにオペレーティングシステムをディスクに書き込み、ホストを再起動します。

ホストインベントリーと central infrastructure management の詳細は、以下を参照してください。

- Central Infrastructure Management サービスの有効化
- Amazon Web Services での Central Infrastructure Management の有効化
- コンソールを使用したホストインベントリーの作成
- コマンドラインインターフェイスを使用したホストインベントリーの作成
- インフラストラクチャー環境用の高度なネットワークの設定
- Discovery Image を使用したホストのホストインベントリーへの追加
- ベアメタルホストのホストインベントリーへの自動追加
- ホストインベントリーの管理
- オンプレミス環境でのクラスターの作成

1.5.3.1. Central Infrastructure Management サービスの有効化

Central Infrastructure Management サービスは multicluster engine Operator とともに提供され、OpenShift Container Platform クラスターをデプロイします。ハブクラスターで MultiClusterHub Operator を有効にすると、central infrastructure management が自動的にデプロイされますが、サービスを手動で有効にする必要があります。

1.5.3.1.1. 前提条件

Central Infrastructure Management サービスを有効にする前に、次の前提条件を確認してください。

- サポートされているバージョンの Red Hat Advanced Cluster Management for Kubernetes を備えた OpenShift Container Platform 4.13 以降にハブクラスターがデプロイされている。

- クラスターを作成するために必要なイメージを取得するためのハブクラスターへのインターネットアクセス (接続済み)、あるいはインターネットへの接続がある内部またはミラーレジストリーへの接続 (非接続) がある。
- ベアメタルプロビジョニングに必要なポートを開く必要があります。OpenShift Container Platform ドキュメントの 必須ポートが開いていることを確認する を参照してください。
- ベアメタルホストのカスタムリソース定義がある。
- OpenShift Container Platform プルシークレット がある。詳細は、**イメージプルシークレットの使用** を参照してください。
- 設定済みのデフォルトのストレージクラスがある。
- 切断された環境の場合のみ、OpenShift Container Platform ドキュメントの ネットワーク遠端のクラスター に関する手順を完了してください。

1.5.3.1.2. ベアメタルホストのカスタムリソース定義の作成

Central Infrastructure Management サービスを有効にする前に、ベアメタルホストのカスタムリソース定義が必要です。

1. 次のコマンドを実行して、ベアメタルホストのカスタムリソース定義がすでに存在するかどうかを確認します。

```
oc get crd baremetalhosts.metal3.io
```

   - ベアメタルホストのカスタムリソース定義がある場合、出力にはリソースが作成された日付が表示されます。
   - リソースがない場合は、次のようなエラーが表示されます。

```
Error from server (NotFound): customresourcedefinitions.apiextensions.k8s.io
"baremetalhosts.metal3.io" not found
```

2. ベアメタルホストのカスタムリソース定義がない場合は、metal3.io_baremetalhosts.yaml ファイルをダウンロードし、次のコマンドを実行することでコンテンツを適用して、リソースを作成します。

```
oc apply -f
```

1.5.3.1.3. Provisioning リソースの作成または変更

Central Infrastructure Management サービスを有効にする前に、**Provisioning** リソースが必要です。

1. 次のコマンドを実行して、**Provisioning** リソースがあるかどうかを確認します。

```
oc get provisioning
```

   - すでに **Provisioning** リソースがある場合は、**Provisioning リソースの変更** に進みます。
   - **Provisioning** リソースがない場合は、**No resources found** というエラーが表示されます。**Provisioning リソースの作成** に進みます。

1.5.3.1.3.1. Provisioning リソースの変更

すでに **Provisioning** リソースがある場合は、ハブクラスターが次のいずれかのプラットフォームにインストールされている場合、リソースを変更する必要があります。

- ベアメタル
- Red Hat OpenStack Platform
- VMware vSphere
- ユーザープロビジョニングインフラストラクチャー (UPI) 方式とプラットフォームは **None** です

ハブクラスターが別のプラットフォームにインストールされている場合は、**非接続環境での central infrastructure management の有効化** または **接続環境での central infrastructure management の有効化** に進みます。

1. **provisioning** リソースを変更し、以下のコマンドを実行してベアメタル Operator がすべての namespace を監視できるようにします。

```
oc patch provisioning provisioning-configuration --type merge -p '{"spec":
{"watchAllNamespaces": true }}'
```

1.5.3.1.3.2. Provisioning リソースの作成

Provisioning リソースがない場合は、次の手順を実行します。

1. 次の YAML コンテンツを追加して、**Provisioning** リソースを作成します。

```
apiVersion: metal3.io/v1alpha1
kind: Provisioning
metadata:
  name: provisioning-configuration
spec:
  provisioningNetwork: "Disabled"
  watchAllNamespaces: true
```

2. 次のコマンドを実行してコンテンツを適用します。

```
oc apply -f
```

1.5.3.1.4. 非接続環境での central infrastructure management の有効化

非接続環境で central infrastructure management を有効にするには、以下の手順を実行します。

1. インフラストラクチャー Operator と同じ namespace に **ConfigMap** を作成し、ミラーレジストリーの **ca-bundle.crt** および **registries.conf** の値を指定します。ファイルの **ConfigMap** は次の例のようになります。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: <mirror-config>
  namespace: multicluster-engine
  labels:
    app: assisted-service
```

```
data:
  ca-bundle.crt: |
    <certificate-content>
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
    [[registry]]
      prefix = ""
      location = "registry.redhat.io/multicluster-engine"
      mirror-by-digest-only = true
      [[registry.mirror]]
      location = "mirror.registry.com:5000/multicluster-engine"
```

unqualified-search-registries のリストにあるレジストリーは、**PUBLIC_CONTAINER_REGISTRIES** 環境変数の認証無視リストに自動的に追加されます。マネージドクラスターのプルシークレットの検証時に、指定されたレジストリーは認証を必要としません。

2. すべての **osImage** リクエストで送信するヘッダーとクエリーパラメーターを表すキーペアを書き込みます。両方のパラメーターが必要ない場合は、ヘッダーのみ、またはクエリーパラメーターのみのキーペアを作成します。

重要: ヘッダーとクエリーパラメーターは、HTTPS を使用する場合にのみ暗号化されます。セキュリティーの問題を避けるために、必ず HTTPS を使用してください。

a. **headers** という名前のファイルを作成し、次の例のような内容を追加します。

```
{
  "header1": "header1value",
  "header2": "header2value",
}
```

b. **query_params** という名前のファイルを作成し、次の例のような内容を追加します。

```
{
  "param1": "value1",
  "param2": "value2",
}
```

1. 次のコマンドを実行して、作成したパラメーターファイルからシークレットを作成します。パラメーターファイルを 1 つだけ作成した場合は、作成しなかったファイルの引数を削除します。

```
oc create secret generic -n multicluster-engine os-images-http-auth --from-file=./query_params --from-file=./headers
```

2. 自己署名証明書またはサードパーティー CA 証明書とともに HTTPS **osImage** を使用する場合は、証明書を **image-service-Additional-ca ConfigMap** に追加します。証明書を作成するには、次のコマンドを実行します。

```
oc -n multicluster-engine create configmap image-service-additional-ca --from-file=tls.crt
```

3. 次の YAML コンテンツを **agent_service_config.yaml** ファイルに保存して、**AgentServiceConfig** カスタムリソースを作成します。

```
apiVersion: agent-install.openshift.io/v1beta1
```

```
kind: AgentServiceConfig
metadata:
 name: agent
spec:
  databaseStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <db_volume_size>
  filesystemStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <fs_volume_size>
  mirrorRegistryRef:
    name: <mirror_config> 1
  unauthenticatedRegistries:
    - <unauthenticated_registry> 2
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size> 3
  OSImageAdditionalParamsRef:
     name: os-images-http-auth
  OSImageCACertRef:
    name: image-service-additional-ca
  osImages:
    - openshiftVersion: "<ocp_version>" 4
      version: "<ocp_release_version>" 5
      url: "<iso_url>" 6
      cpuArchitecture: "x86_64"
```

1 1 **mirror_config** は、ミラーレジストリー設定の詳細が含まれる **ConfigMap** の名前に置き換えます。

2 認証を必要としないミラーレジストリーを使用している場合は、オプションの **unauthenticated_registry** パラメーターを含めます。このリストのエントリーは検証されず、プルシークレットにエントリーを含める必要はありません。

3 **img_volume_size** を **imageStorage** フィールドのボリュームのサイズに置き換えます (例: オペレーティングシステムイメージごとに **10Gi**)。最小値は **10Gi** ですが、推奨される値は **50Gi** 以上です。この値は、クラスターのイメージに割り当てられるストレージの量を指定します。実行中の Red Hat Enterprise Linux CoreOS の各インスタンスに 1 GB のイメージストレージを許可する必要があります。Red Hat Enterprise Linux CoreOS のクラスターおよびインスタンスが多数ある場合は、より高い値を使用する必要がある場合があります。

4 **ocp_version** は、インストールする OpenShift Container Platform バージョンに置き換えます (例: **4.13**)。

5 **ocp_release_version** は、特定のインストールバージョン (例: **49.83.2021032516400**) に置き換えます。

6 **iso_url** は、ISO URL に置き換えます (例: **https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.13/4.13.3/rhcos-4.13.3-x86_64-live.x86_64.iso**)。その他の値は 4.12.3 の依存関係 にあります。

自己署名証明書またはサードパーティー CA 証明書を持つ HTTPS **osImage** を使用している場合は、**OSImageCACertRef** 仕様でその証明書を参照してください。

重要: 遅延バインディング機能を使用しており、**AgentServiceConfig** カスタムリソースの **spec.osImages** リリースがバージョン 4.13 以降である場合、クラスターの作成時に使用する OpenShift Container Platform リリースイメージはバージョン 4.13 以降である必要があります。バージョン 4.13 以降の Red Hat Enterprise Linux CoreOS イメージは、バージョン 4.13 より前のイメージとは互換性がありません。

assisted-service デプロイメントおよび **assisted-image-service** デプロイメントをチェックし、その Pod が準備ができており実行中であることを確認することで、中央インフラストラクチャー管理サービスが正常であることを確認できます。

1.5.3.1.5. 接続環境での central infrastructure management の有効化

接続環境で中央インフラストラクチャー管理を有効にするには、以下の YAML コンテンツを **agent_service_config.yaml** ファイルに保存して、**AgentServiceConfig** カスタムリソースを作成します。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
 name: agent
spec:
  databaseStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <db_volume_size> 1
  filesystemStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <fs_volume_size> 2
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size> 3
```

1 **db_volume_size** は **databaseStorage** フィールドのボリュームサイズに置き換えます (例: **10Gi**)。この値は、クラスターのデータベーステーブルやデータベースビューなどのファイルを格納するために割り当てられるストレージの量を指定します。必要な最小値は **1Gi** です。クラスターが多い場合は、より高い値を使用する必要がある場合があります。

2 **fs_volume_size** は **filesystemStorage** フィールドのボリュームのサイズに置き換えます (例: クラスターごとに **200M**、サポートされる OpenShift Container Platform バージョンごとに **2-3Gi**)。必

3 **img_volume_size** を **imageStorage** フィールドのボリュームのサイズに置き換えます (例: オペレーティングシステムイメージごとに **10Gi**)。最小値は **10Gi** ですが、推奨される値は **50Gi** 以上です。この値は、クラスターのイメージに割り当てられるストレージの量を指定します。実行中の Red Hat Enterprise Linux CoreOS の各インスタンスに 1 GB のイメージストレージを許可する必要があります。Red Hat Enterprise Linux CoreOS のクラスターおよびインスタンスが多数ある場合は、より高い値を使用する必要がある場合があります。

central infrastructure management サービスが設定されている。**assisted-service** と **assisted-image-service** デプロイメントをチェックして、Pod の準備ができ、実行されていることを確認して、正常性を検証できます。

1.5.3.1.6. 関連情報

- ゼロタッチプロビジョニングの詳細は、OpenShift Container Platform ドキュメントの ネットワーク遠端のクラスター を参照してください。
- イメージプルシークレットの使用 を参照してください。

1.5.3.2. Amazon Web Services での Central Infrastructure Management の有効化

Amazon Web Services でハブクラスターを実行していて、central infrastructure management サービスを有効にする場合は、central infrastructure management を central infrastructure management の有効化 後に、次の手順を実行します。

1. ハブクラスターにログインしていることを確認し、次のコマンドを実行して、**assisted-image-service** で設定された一意のドメインを見つけます。

```
oc get routes --all-namespaces | grep assisted-image-service
```

ドメインは **assisted-image-service-multicluster-engine.apps.<yourdomain>.com** のようになります。

2. ハブクラスターにログインしていることを確認し、**NLB type** パラメーターを使用して一意のドメインで新しい **IngressController** を作成します。以下の例を参照してください。

```
apiVersion: operator.openshift.io/v1
kind: IngressController
metadata:
  name: ingress-controller-with-nlb
  namespace: openshift-ingress-operator
spec:
  domain: nlb-apps.<domain>.com
  routeSelector:
     matchLabels:
       router-type: nlb
  endpointPublishingStrategy:
    type: LoadBalancerService
    loadBalancer:
      scope: External
      providerParameters:
```

```
      type: AWS
      aws:
        type: NLB
```

3. **nlb-apps.<domain>.com** の **<domain>** を **<yourdomain>** に置き換えて、**IngressController** の **domain** パラメーターに **<yourdomain>** を追加します。

4. 次のコマンドを実行して、新しい **IngressController** を適用します。

```
oc apply -f ingresscontroller.yaml
```

5. 次の手順を実行して、新しい **IngressController** の **spec.domain** パラメーターの値が既存の **IngressController** と競合していないことを確認します。

   a. 次のコマンドを実行して、すべての **IngressController** を一覧表示します。

```
oc get ingresscontroller -n openshift-ingress-operator
```

   b. 先ほど作成した **ingress-controller-with-nlb** を除く各 **IngressControllers** で次のコマンドを実行します。

```
oc edit ingresscontroller <name> -n openshift-ingress-operator
```

   spec.domain レポートが見つからない場合は、**nlb-apps.<domain>.com** を除く、クラスターで公開されているすべてのルートに一致するデフォルトドメインを追加します。

   spec.domain レポートが提供されている場合は、指定された範囲から **nlb-apps.<domain>.com** ルートが除外されていることを確認してください。

6. 次のコマンドを実行して、**assisted-image-service** ルートを編集し、**nlb-apps** の場所を使用します。

```
oc edit route assisted-image-service -n <namespace>
```

   デフォルトの namespace は、multicluster engine operator をインストールした場所です。

7. 次の行を **assisted-image-service** ルートに追加します。

```
metadata:
  labels:
    router-type: nlb
  name: assisted-image-service
```

8. **assisted-image-service** ルートで、**spec.host** の URL 値を見つけます。URL は次の例のようになります。

```
assisted-image-service-multicluster-engine.apps.<yourdomain>.com
```

9. URL 内の **apps** を **nlb-apps** に置き換えて、新しい **IngressController** で設定されたドメインと一致させます。

10. central infrastructure management サービスが Amazon Web Services で有効になっていることを確認するには、次のコマンドを実行して Pod が正常であることを確認します。

```
oc get pods -n multicluster-engine | grep assist
```

11. 新しいホストインベントリーを作成し、ダウンロード URL が新しい **nlb-apps** URL を使用していることを確認します。

1.5.3.3. コンソールを使用したホストインベントリーの作成

ホストインベントリー (インフラストラクチャー環境) を作成して、OpenShift Container Platform クラスターをインストールできる物理マシンまたは仮想マシンを検出できます。

1.5.3.3.1. 前提条件

- central infrastructure management サービスを有効にする。詳細は、**Central Infrastructure Management サービスの有効化** を参照してください。

1.5.3.3.2. ホストインベントリーの作成

コンソールを使用してホストインベントリーを作成するには、次の手順を実行します。

1. コンソールから、**Infrastructure** > **Host inventory** に移動して、**Create infrastructure environment** をクリックします。
2. 次の情報をホストインベントリー設定に追加します。
   - 名前: インフラストラクチャー環境の一意の名前。コンソールを使用してインフラストラクチャー環境を作成すると、選択した名前で **InfraEnv** リソースの新しい namespace も作成されます。コマンドラインインターフェイスを使用して **InfraEnv** リソースを作成し、コンソールでリソースを監視する場合は、namespace と **InfraEnv** に同じ名前を使用します。
   - ネットワークタイプ: インフラストラクチャー環境に追加するホストが DHCP または静的ネットワークを使用するかどうかを指定します。静的ネットワーク設定には、追加の手順が必要です。
   - 場所: ホストの地理的な場所を指定します。地理的なロケーションを使用して、ホストが配置されているデータセンターを定義できます。
   - ラベル: このインフラストラクチャー環境で検出されたホストにラベルを追加できるオプションのフィールド。指定した場所はラベルのリストに自動的に追加されます。
   - インフラストラクチャープロバイダーの認証情報: インフラストラクチャープロバイダーの認証情報を選択すると、プルシークレットおよび SSH 公開鍵フィールドに認証情報内の情報が自動的に入力されます。詳細は、**オンプレミス環境の認証情報の作成** を参照してください。
   - プルシークレット: OpenShift Container Platform リソースへのアクセスを可能にする OpenShift Container Platform プルシークレット。インフラストラクチャープロバイダーの認証情報を選択した場合、このフィールドには自動的に入力されます。
   - SSH 公開鍵: ホストとのセキュアな通信を可能にする SSH キー。これを使用してホストに接続し、トラブルシューティングを行うことができます。クラスターをインストールした後は、SSH 鍵を使用してホストに接続できなくなります。通常、鍵は **id_rsa.pub** ファイルにあります。デフォルトのファイルパスは **~/.ssh/id_rsa.pub** です。SSH 公開鍵の値を含むインフラストラクチャープロバイダーの認証情報を選択した場合、このフィールドには自動的に入力されます。

- ホストのプロキシー設定を有効にする場合は、有効にする設定を選択し、次の情報を入力します。
  - HTTP プロキシー URL: HTTP リクエストのプロキシーの URL。
  - HTTPS プロキシー URL: HTTP リクエストのプロキシーの URL。URL は HTTP で始まる必要があります。HTTPS はサポートされていません。値を指定しない場合、デフォルトで HTTP 接続と HTTPS 接続の両方に HTTP プロキシー URL が使用されます。
  - プロキシーなしドメイン: プロキシーを使用しないドメインのコンマ区切りのリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 必要に応じて、NTP プールまたはサーバーの IP 名またはドメイン名のコンマ区切りリストを指定して、独自のネットワークタイムプロトコル (NTP) ソースを追加します。

コンソールでは使用できない詳細な設定オプションが必要な場合は、**コマンドラインインターフェイスを使用したホストインベントリーの作成** に進みます。

高度な設定オプションが必要ない場合は、必要に応じて静的ネットワークの設定を続行し、インフラストラクチャー環境へのホストの追加を開始できます。

1.5.3.3.3. ホストインベントリーへのアクセス

ホストインベントリーにアクセスするには、コンソールで **Infrastructure** > **Host inventory** を選択します。一覧からインフラストラクチャー環境を選択し、詳細とホストを表示します。

1.5.3.3.4. 関連情報

- Central Infrastructure Management サービスの有効化 を参照してください。
- オンプレミス環境の認証情報の作成 を参照してください。
- コマンドラインインターフェイスを使用したホストインベントリーの作成 を参照してください。
- ベアメタル上で Hosted control plane を設定するプロセスの一部としてこの手順を完了した場合は、次の手順を完了してください。
  - Discovery Image を使用したホストのホストインベントリーへの追加
  - ベアメタルホストのホストインベントリーへの自動追加

1.5.3.4. コマンドラインインターフェイスを使用したホストインベントリーの作成

ホストインベントリー (インフラストラクチャー環境) を作成して、OpenShift Container Platform クラスターをインストールできる物理マシンまたは仮想マシンを検出できます。自動化されたデプロイメントや、以下の高度な設定オプションには、コンソールの代わりにコマンドラインインターフェイスを使用します。

- 検出されたホストを既存のクラスター定義に自動的にバインドする
- Discovery Image の Ignition 設定をオーバーライドする
- iPXE の動作を制御する
- Discovery Image のカーネル引数を変更する

- 検出フェーズ中にホストに信頼させる追加の証明書を渡す
- 最新バージョンのデフォルトオプションではない、テスト用に起動する Red Hat CoreOS バージョンを選択する

1.5.3.4.1. 前提条件

- central infrastructure management サービスを有効にする。詳細は、**Central Infrastructure Management サービスの有効化** を参照してください。

1.5.3.4.2. ホストインベントリーの作成

コマンドラインインターフェイスを使用してホストインベントリー (インフラストラクチャー環境) を作成するには、次の手順を実行します。

1. 次のコマンドを実行して、ハブクラスターにログインします。

```
oc login
```

2. リソースの namespace を作成します。
   a. **namespace.yaml** ファイルを作成し、以下の内容を追加します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: <your_namespace> 1
```

1 コンソールでインベントリーを監視するには、namespace とインフラストラクチャー環境に同じ名前を使用します。

   b. 以下のコマンドを実行して YAML コンテンツを適用します。

```
oc apply -f namespace.yaml
```

3. OpenShift Container Platform プルシークレット を含む **Secret** カスタムリソースを作成します。
   a. **pull-secret.yaml** ファイルを作成し、以下の内容を追加します。

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret 1
  namespace: <your_namespace>
stringData:
  .dockerconfigjson: <your_pull_secret> 2
```

1 namesapce を追加します。

2 プルシークレットを追加します。

b. 以下のコマンドを実行して YAML コンテンツを適用します。

```
oc apply -f pull-secret.yaml
```

4. インフラストラクチャー環境を作成します。

a. **infra-env.yaml** ファイルを作成し、以下の内容を追加します。必要に応じて値を置き換えます。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: myinfraenv
  namespace: <your_namespace>
spec:
  proxy:
    httpProxy: <http://user:password@ipaddr:port>
    httpsProxy: <http://user:password@ipaddr:port>
    noProxy:
  additionalNTPSources:
  sshAuthorizedKey:
  pullSecretRef:
    name: <name>
  agentLabels:
    <key>: <value>
  nmStateConfigLabelSelector:
    matchLabels:
      <key>: <value>
  clusterRef:
    name: <cluster_name>
    namespace: <project_name>
  ignitionConfigOverride: '{"ignition": {"version": "3.1.0"}, …}'
  cpuArchitecture: x86_64
  ipxeScriptType: DiscoveryImageAlways
  kernelArguments:
    - operation: append
      value: audit=0
  additionalTrustBundle: <bundle>
  osImageVersion: <version>
```

表1.4 InfraEnv のフィールドの表

フィールド	任意または必須	説明
proxy	任意	**InfraEnv** リソースを使用するエージェントとクラスターのプロキシー設定を定義します。**proxy** の値を設定しない場合、エージェントはプロキシーを使用するように設定されません。

フィールド	任意または必須	説明
httpProxy	任意	HTTP リクエストのプロキシーの URLURL は **http** で開始する必要があります。HTTPS はサポートされていません。
httpsProxy	任意	HTTP リクエストのプロキシーの URLURL は **http** で開始する必要があります。HTTPS はサポートされていません。
noProxy	任意	プロキシーを使用しない場合は、コンマで区切られたドメインおよび CIDR のリスト。
additionalNTPSources	任意	すべてのホストに追加するネットワークタイムプロトコル (NTP) ソース (ホスト名または IP) のリスト。これらは、DHCP などの他のオプションを使用して設定された NTP ソースに追加されます。
sshAuthorizedKey	任意	検出フェーズ中のデバッグに使用するためにすべてのホストに追加される SSH 公開鍵。検出フェーズは、ホストが検出イメージを起動するときです。
name	必須	プルシークレットが含まれる Kubernetes シークレットの名前。
agentLabels	任意	**InfraEnv** で検出されたホストを表す **Agent** リソースに自動的に追加されるラベル。キーと値を必ず追加してください。

フィールド	任意または必須	説明
nmStateConfigLabelSelector	任意	ホストの静的 IP、ブリッジ、ボンディングなどの高度なネットワーク設定を統合します。ホストネットワーク設定は、選択したラベルを持つ 1 つ以上の **NMStateConfig** リソースで指定されます。**nmStateConfigLabelSelector** プロパティーは、選択したラベルと一致する Kubernetes ラベルセレクターです。このラベルセレクターに一致するすべての **NMStateConfig** ラベルのネットワーク設定は、Discovery Image に含まれています。起動時に、各ホストは各設定をそのネットワークインターフェイスと比較し、適切な設定を適用します。高度なネットワーク設定の詳細は、**ホストインベントリーの高度なネットワークの設定** セクションへのリンクを参照してください。
clusterRef	任意	スタンドアロンのオンプレミスクラスターを記述する既存の **ClusterDeployment** リソースを参照します。デフォルトでは設定されません。**clusterRef** が設定されていない場合は、後でホストを 1 つ以上のクラスターにバインドできます。あるクラスターからホストを削除し、別のクラスターに追加できます。**clusterRef** が設定されている場合、**InfraEnv** で検出されたすべてのホストが、指定されたクラスターに自動的にバインドされます。クラスターがまだインストールされていない場合は、検出されたすべてのホストがそのインストールに含まれます。クラスターがすでにインストールされている場合は、検出されたすべてのホストが追加されます。

フィールド	任意または必須	説明
ignitionConfigOverride	任意	ファイルの追加など、Red Hat CoreOS ライブイメージの Ignition 設定を変更します。必要に応じて **ignitionConfigOverride** のみを使用してください。クラスターのバージョンに関係なく、ignition バージョン 3.1.0 を使用する必要があります。
cpuArchitecture	任意	サポートされている CPU アーキテクチャー x86_64、aarch64、ppc64le、または s390x のいずれかを選択します。デフォルト値は x86_64 です。
ipxeScriptType	任意	起動に iPXE を使用している場合に、デフォルト値である **DiscoveryImageAlways** に設定すると、イメージサービスが常に iPXE スクリプトを提供するようになります。その結果、ホストがネットワーク検出イメージから起動します。値を **BootOrderControl** に設定すると、イメージサービスがホストの状態に応じて iPXE スクリプトを返すタイミングを決定するようになります。その結果、ホストがプロビジョニングされていてクラスターの一部になっている場合、ホストはディスクから起動します。
kernelArguments	任意	Discovery Image の起動時にカーネル引数を変更できます。**operation** に使用できる値は、**append**、**replace**、または **delete** です。

フィールド	任意または必須	説明
additionalTrustBundle	任意	PEM エンコードされた X.509 証明書バンドル。通常、ホストが再暗号化中間者 (MITM) プロキシーを備えたネットワーク内にある場合、またはコンテナーイメージレジストリーなど、他の目的でホストが証明書を信頼しなければならない場合に必要です。**InfraEnv** によって検出されたホストは、このバンドル内の証明書を信頼します。**InfraEnv** によって検出されたホストから作成されたクラスターは、このバンドル内の証明書も信頼します。
osImageVersion	任意	**InfraEnv** に使用する Red Hat CoreOS イメージのバージョン。バージョンが **AgentServiceConfig.spec.osImages** またはデフォルトの OS イメージのリストで指定された OS イメージを参照していることを確認してください。各リリースには、特定の Red Hat CoreOS イメージバージョンのセットがあります。**OSImageVersion** は、OS イメージリストの OpenShift Container Platform バージョンと一致する必要があります。**OSImageVersion** と **ClusterRef** を同時に指定できません。デフォルトでは存在しない別のバージョンの Red Hat CoreOS イメージを使用する場合は、**AgentServiceConfig.spec.osImages** でそのバージョンを指定して、手動で追加する必要があります。バージョンの追加に関する詳細は、**中央インフラストラクチャー管理サービスの有効化** を参照してください。

a. 以下のコマンドを実行して YAML コンテンツを適用します。

```
oc apply -f infra-env.yaml
```

b. ホストインベントリーが作成されたことを確認するには、次のコマンドでステータスを確認します。

```
oc describe infraenv myinfraenv -n <your_namespace>
```

重要なプロパティーのリストは次のとおりです。

- **conditions**: イメージが正常に作成されたかどうかを示す標準の Kubernetes 条件。
- **isoDownloadURL**: Discovery Image をダウンロードするための URL。
- **createdTime**: イメージが最後に作成された時刻。**InfraEnv** を変更する場合は、新しいイメージをダウンロードする前にタイムスタンプが更新されていることを確認してください。

注: InfraEnv リソースを変更する場合は、**createdTime** プロパティーを確認して、**InfraEnv** が新しい Discovery Image を作成したことを確認してください。すでにホストを起動している場合は、最新の Discovery Image でホストを再起動します。

必要に応じて静的ネットワークを設定し、インフラストラクチャー環境へのホストの追加を開始することで続行できます。

1.5.3.4.3. 関連情報

- Central Infrastructure Management サービスの有効化 を参照してください。

1.5.3.5. インフラストラクチャー環境用の高度なネットワークの設定

1 つのインターフェイスで DHCP 以外のネットワークを必要とするホストの場合は、高度なネットワークを設定する必要があります。必要な設定には、1 つ以上のホストのネットワークを記述する **NMStateConfig** リソースの 1 つ以上のインスタンスの作成が含まれます。

各 **NMStateConfig** リソースには、**InfraEnv** リソースの **nmStateConfigLabelSelector** に一致するラベルが含まれている必要があります。**nmStateConfigLabelSelector** の詳細は、**コマンドラインインターフェイスを使用したホストインベントリーの作成** を参照してください。

Discovery Image には、参照されているすべての **NMStateConfig** リソースで定義されたネットワーク設定が含まれています。起動後、各ホストは各設定をそのネットワークインターフェイスと比較し、適切な設定を適用します。

1.5.3.5.1. 前提条件

- central infrastructure management サービスを有効にする。
- ホストインベントリーを作成する。

1.5.3.5.2. コマンドラインインターフェイスを使用した高度なネットワークの設定

コマンドラインインターフェイスを使用してインフラストラクチャー環境の詳細ネットワークを設定するには、以下の手順を実行します。

1. **nmstateconfig.yaml** という名前のファイルを作成し、以下のテンプレートのようなコンテンツを追加します。必要に応じて値を置き換えます。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: mynmstateconfig
  namespace: <your-infraenv-namespace>
  labels:
    some-key: <some-value>
spec:
```

```
config:
  interfaces:
    - name: eth0
      type: ethernet
      state: up
      mac-address: 02:00:00:80:12:14
      ipv4:
        enabled: true
        address:
          - ip: 192.168.111.30
            prefix-length: 24
        dhcp: false
    - name: eth1
      type: ethernet
      state: up
      mac-address: 02:00:00:80:12:15
      ipv4:
        enabled: true
        address:
          - ip: 192.168.140.30
            prefix-length: 24
        dhcp: false
  dns-resolver:
    config:
      server:
        - 192.168.126.1
  routes:
    config:
      - destination: 0.0.0.0/0
        next-hop-address: 192.168.111.1
        next-hop-interface: eth1
        table-id: 254
      - destination: 0.0.0.0/0
        next-hop-address: 192.168.140.1
        next-hop-interface: eth1
        table-id: 254
interfaces:
  - name: "eth0"
    macAddress: "02:00:00:80:12:14"
  - name: "eth1"
    macAddress: "02:00:00:80:12:15"
```

表1.5 NMStateConfig のフィールドの表

フィールド	任意または必須	説明
name	必須	設定しているホストに関連した名前を使用してください。
namespace	必須	namespace は、**InfraEnv** リソースの namespace と一致する必要があります。

フィールド	任意または必須	説明
some-key	必須	**nmStateConfigLabelSelector** に一致する 1 つ以上のラベルを **InfraEnv** リソースに追加します。
config	任意	**NMstate** 形式のネットワーク設定について説明します。形式の指定と追加の例については、Declarative Network API を参照してください。この設定は、ホストごとに 1 つの **NMStateConfig** リソースがある単一のホストに適用することも、単一の **NMStateConfig** リソースで複数のホストのインターフェイスを記述することもできます。
interfaces	任意	指定された **NMstate** 設定で見つかったインターフェイス名とホストで見つかった MAC アドレスの間のマッピングを記述します。マッピングがホスト上に存在する物理インターフェイスを使用していることを確認してください。たとえば、**NMState** 設定でボンドまたは VLAN が定義されている場合、マッピングには親インターフェイスのエントリーのみが含まれます。マッピングには次の目的があります。* ホスト上のインターフェイス名と一致しないインターフェイス名を設定で使用できるようにします。オペレーティングシステムによりインターフェイス名 (これは予測できない) が選択されるので、便利な場合があります。* 起動後に検索する MAC アドレスをホストに指示し、正しい **NMstate** 設定を適用します。

注: イメージサービスは、**InfraEnv** プロパティーを更新するか、そのラベルセレクターに一致する **NMStateConfig** リソースを変更すると、新しいイメージを自動的に作成します。**InfraEnv** リソースの作成後に **NMStateConfig** リソースを追加する場合は、**InfraEnv** の **createdTime** プロパティーを確認して、**InfraEnv が** 新しい Discovery Image を作成していることを確認してください。すでにホストを起動している場合は、最新のDiscovery Image でホストを再起動します。

1. 以下のコマンドを実行して YAML コンテンツを適用します。

```
oc apply -f nmstateconfig.yaml
```

1.5.3.5.3. 関連情報

- コマンドラインインターフェイスを使用したホストインベントリーの作成 を参照してください。
- Declarative Network API を参照してください。

1.5.3.6. Discovery Image を使用したホストのホストインベントリーへの追加

ホストインベントリー (インフラストラクチャー環境) を作成した後、ホストを検出してインベントリーに追加できます。インベントリーにホストを追加するには、ISO をダウンロードし、各サーバーにアタッチする方法を選択します。たとえば、仮想メディアを使用するか、ISO を USB ドライブに書き込むことで、ISO をダウンロードできます。

重要: インストールが失敗しないようにするには、インストールプロセス中は Discovery ISO メディアをデバイスに接続したままにし、各ホストがデバイスから 1 回起動するように設定します。

1.5.3.6.1. 前提条件

- central infrastructure management サービスを有効にする。詳細は、**Central Infrastructure Management サービスの有効化** を参照してください。
- ホストインベントリーを作成する。詳細は、**コンソールを使用したホストインベントリーの作成** を参照してください。

1.5.3.6.2. コンソールを使用したホストの追加

以下の手順を実行して ISO をダウンロードします。

1. コンソールで **Infrastructure** > **Host inventory** を選択します。
2. 一覧からお使いのインフラストラクチャー環境を選択します。
3. **Add hosts** をクリックし、**With Discovery ISO** を選択します。

ISO をダウンロードするための URL が表示されます。ブートされたホストがホストインベントリーテーブルに表示されます。ホストが表示されるまでに数分かかる場合があります。使用する前に、各ホストを承認する必要があります。**Actions** をクリックして **Approve** を選択し、インベントリーテーブルからホストを選択できます。

1.5.3.6.3. コマンドラインインターフェイスを使用したホストの追加

ISO をダウンロードするための URL は、**InfraEnv** リソースのステータスの **isoDownloadURL** プロパティーで確認できます。**InfraEnv** リソースの詳細は、コマンドラインインターフェイスを使用したホストインベントリーの作成を参照してください。

起動したホストごとに、同じ namespace に **Agent** リソースを作成します。使用する前に、各ホストを承認する必要があります。

1.5.3.6.4. 関連情報

- Central Infrastructure Management サービスの有効化 を参照してください。
- コンソールを使用したホストインベントリーの作成

- コマンドラインインターフェイスを使用したホストインベントリーの作成 を参照してください。

1.5.3.7. ベアメタルホストのホストインベントリーへの自動追加

ホストインベントリー (インフラストラクチャー環境) を作成した後、ホストを検出してインベントリーに追加できます。各ホストの **BareMetalHost** リソースおよび関連する BMC シークレットを作成することで、ベアメタル Operator が各ベアメタルホストのベースボード管理コントローラー (BMC) と通信できるようにすることで、インフラストラクチャー環境の Discovery Image の起動を自動化できます。自動化は、インフラストラクチャー環境を参照する **BareMetalHost** のラベルによって設定されます。

自動化により以下のアクションが実行されます。

- インフラストラクチャー環境で表される Discovery Image を使用して、各ベアメタルホストを起動します。
- インフラストラクチャー環境または関連するネットワーク設定が更新された場合に、各ホストを最新の Discovery Image で再起動します。
- 検出時に各 **Agent** リソースを対応する **BareMetalHost** リソースに関連付けます。
- **BareMetalHost** からの情報 (ホスト名、ロール、インストールディスクなど) に基づいて **Agent** リソースのプロパティーを更新します。
- **Agent** をクラスターノードとして使用することを承認します。

1.5.3.7.1. 前提条件

- central infrastructure management サービスを有効にする。
- ホストインベントリーを作成する。

1.5.3.7.2. コンソールを使用したベアメタルホストの追加

コンソールを使用してベアメタルホストをホストインベントリーに自動的に追加するには、次の手順を実行します。

1. コンソールで **Infrastructure** > **Host inventory** を選択します。
2. 一覧からお使いのインフラストラクチャー環境を選択します。
3. **Add hosts** をクリックし、**With BMC Form** を選択します。
4. 必要な情報を追加し、**Create** をクリックします。

1.5.3.7.3. コマンドラインインターフェイスを使用したベアメタルホストの追加

コマンドラインインターフェイスを使用してベアメタルホストをホストインベントリーに自動的に追加するには、以下の手順を実施します。

1. 次の YAML コンテンツを適用し、必要に応じて値を置き換えて、BMC シークレットを作成します。

```
apiVersion: v1
kind: Secret
metadata:
```

```
  name: <bmc-secret-name>
  namespace: <your_infraenv_namespace> 1
type: Opaque
data:
  username: <username>
  password: <password>
```

1 namespace は **InfraEnv** の namespace と同じである必要があります。

2. 以下の YAML コンテンツを適用し、必要に応じて値を置き換えてベアメタルホストを作成します。

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: <bmh-name>
  namespace: <your_infraenv_namespace> 1
  annotations:
    inspect.metal3.io: disabled
  labels:
    infraenvs.agent-install.openshift.io: <your-infraenv> 2
spec:
  online: true
  automatedCleaningMode: disabled 3
  bootMACAddress: <your-mac-address> 4
  bmc:
    address: <machine-address> 5
    credentialsName: <bmc-secret-name> 6
  rootDeviceHints:
    deviceName: /dev/sda 7
```

1 namespace は **InfraEnv** の namespace と同じである必要があります。

2 この名前は **InfrEnv** の名前と一致し、同じ namespace に存在する必要があります。

3 値を設定しない場合、**metadata** の値が自動的に使用されます。

4 MAC アドレスがホスト上のいずれかのインターフェイスの MAC アドレスと一致することを確認してください。

5 BMC のアドレスを使用します。詳細は **帯域外管理 IP アドレスのポートアクセス** を参照してください。

6 **credentialsName** の 値が、作成した BMC シークレットの名前と一致していることを確認してください。

7 **オプション:** インストールディスクを選択します。利用可能なルートデバイスのヒントについては、**BareMetalHost spec** を参照してください。ホストが Discovery Image で起動され、対応する **Agent** リソースが作成された後、このヒントに従ってインストールディスクが設定されます。

ホストの電源をオンにすると、イメージのダウンロードが開始されます。これには数分かかる場合があります。ホストが検出されると、**Agent** カスタムリソースが自動的に作成されます。

1.5.3.7.4. コマンドラインインターフェイスを使用したマネージドクラスターノードの削除

マネージドクラスターからマネージドクラスターノードを削除するには、OpenShift Container Platform バージョン 4.13 以降を実行しているハブクラスターが必要です。ノードの起動に必要な静的ネットワーク設定が利用可能である必要があります。エージェントとベアメタルホストを削除するときに、**NMStateConfig** リソースを削除しないようにしてください。

1.5.3.7.4.1. ベアメタルホストを使用したマネージドクラスターノードの削除

ハブクラスター上にベアメタルホストがあり、マネージドクラスターからマネージドクラスターノードを削除する場合は、次の手順を実行します。

1. 削除するノードの **BareMetalHost** リソースに次のアノテーションを追加します。

```
bmac.agent-install.openshift.io/remove-agent-and-node-on-delete: true
```

2. 次のコマンドを実行して、**BareMetalHost** リソースを削除します。**<bmh-name>** は、**BareMetalHost** の名前に置き換えます。

```
oc delete bmh <bmh-name>
```

1.5.3.7.4.2. ベアメタルホストを使用しないマネージドクラスターノードの削除

ハブクラスターにベアメタルホストがなく、マネージドクラスターからマネージドクラスターノードを削除する場合は、OpenShift Container Platform ドキュメントの ノードの削除 手順に従ってください。

1.5.3.7.5. 関連情報

- ゼロタッチプロビジョニングの詳細は、OpenShift Container Platform ドキュメントの ネットワーク遠端のクラスター を参照してください。
- ベアメタルホストを使用するために必要なポートの詳細は、OpenShift Container Platform ドキュメントの 帯域外管理 IP アドレスのポートアクセス を参照してください。
- ルートデバイスのヒントについては、OpenShift Container Platform ドキュメントの BareMetalHost 仕様 を参照してください。
- イメージプルシークレットの使用 を参照してください。
- オンプレミス環境の認証情報の作成 を参照してください。
- コンピュートマシンのスケーリングの詳細は、OpenShift Container Platform ドキュメントの コンピュートマシンセットの手動によるスケーリング を参照してください。

1.5.3.8. ホストインベントリーの管理

コンソールまたはコマンドラインインターフェイスを使用して、**Agent** リソースの編集、ホストインベントリーの管理、既存のホストの編集が可能です。

1.5.3.8.1. コンソールを使用したホストインベントリーの管理

Discovery ISO で正常に起動した各ホストは、ホストインベントリーの行として表示されます。コンソールを使用してホストを編集および管理できます。ホストを手動で起動し、ベアメタル Operator の自動化を使用していない場合は、使用する前にコンソールでホストを承認する必要があります。OpenShift ノードとしてインストールできるホストが **Available** ステータスになっています。

1.5.3.8.2. コマンドラインインターフェイスを使用したホストインベントリーの管理

Agent リソースは、各ホストを表します。**Agent** リソースで次のプロパティーを設定できます。

- **clusterDeploymentName**
  このプロパティーは、クラスターにノードとしてインストールする場合に使用する **ClusterDeployment** の namespace および名前に設定します。

- **任意: role**
  クラスター内のホストのロールを設定します。使用できる値は、**master**、**worker**、および **auto-assign** です。デフォルト値は **auto-assign** です。

- **hostname**
  ホストのホスト名を設定します。ホストに有効なホスト名が自動的に割り当てられる場合は任意です (例: DHCP を使用)。

- **approved**
  ホストを OpenShift ノードとしてインストールできるかどうかを示します。このプロパティーは、デフォルト値が **False** のブール値です。ホストを手動で起動しており、ベアメタル Operator の自動化を使用していない場合は、ホストをインストールする前にこのプロパティーを **True** に設定する必要があります。

- **installation_disk_id**
  ホストのインベントリーに表示されるインストールディスクの ID。

- **installerArgs**
  ホストの **coreos-installer** 引数のオーバーライドが含まれる JSON 形式の文字列。このプロパティーを使用して、カーネル引数を変更できます。構文例を以下に示します。

  ```
  ["--append-karg",
  "ip=192.0.2.2::192.0.2.254:255.255.255.0:core0.example.com:enp1s0:none", "--save-partindex", "4"]
  ```

- **ignitionConfigOverrides**
  ホストの Ignition 設定のオーバーライドが含まれる JSON 形式の文字列。このプロパティーを使用して、ignition を使用してファイルをホストに追加できます。構文例を以下に示します。

  ```
  {"ignition": "version": "3.1.0"}, "storage": {"files": [{"path": "/tmp/example", "contents": {"source": "data:text/plain;base64,aGVscGltdHJhcHBlZGluYXN3YWdnZXJzcGVj"}}]}}
  ```

- **nodeLabels**
  ホストのインストール後にノードに適用されるラベルのリスト。

Agent リソースの **status** には、以下のプロパティーがあります。

- **role**
  クラスター内のホストのロールを設定します。これまでに **Agent** リソースで **role** をしたことがある場合は、その値が **status** に表示されます。

- **inventory**
  ホスト上で実行されているエージェントが検出するホストプロパティーが含まれます。

- **progress**
  ホストのインストールの進行状況。

- **ntpSources**
  ホストの設定済みの Network Time Protocol (NTP)ソース。

- **conditions**
  次の標準 Kubernetes 条件 (**True** または **False** 値) が含まれます。

  - SpecSynced: 指定されたすべてのプロパティーが正常に適用される場合は **True**。何らかのエラーが発生した場合は、**False**。
  - connected: インストールサービスへのエージェント接続が禁止されていない場合は **True**。エージェントがしばらくの間インストールサービスに接続していない場合は **False**。
  - RequirementsMet: ホストがインストールを開始する準備ができている場合は **True**。
  - Validated: すべてのホスト検証に合格した場合は **True**。
  - installed: ホストが OpenShift ノードとしてインストールされている場合は **True**。
  - Bound: ホストがクラスターにバインドされている場合は **True**。
  - Cleanup: **Agent** リソースの削除リクエストが失敗した場合は **False**。

- **debugInfo**
  インストールログおよびイベントをダウンロードするための URL が含まれています。

- **validationsInfo**
  ホストの検出後にインストールが成功したことを確認するためにエージェントが実行する検証の情報が含まれます。値が **False** の場合は、トラブルシューティングを行ってください。

- **installation_disk_id**
  ホストのインベントリーに表示されるインストールディスクの ID。

1.5.3.8.3. 関連情報

- ホストインベントリーへのアクセス を参照してください。
- coreos-installer install を参照してください。

1.5.4. クラスター作成

multicluster engine operator を使用して、クラウドプロバイダー全体で Red Hat OpenShift Container Platform クラスターを作成する方法を説明します。

multicluster engine operator は、OpenShift Container Platform で提供される Hive Operator を使用して、オンプレミスクラスターと Hosted control plane を除くすべてのプロバイダーのクラスターをプロビジョニングします。オンプレミスクラスターをプロビジョニングする場合、multicluster engine Operator は OpenShift Container Platform で提供される Central Infrastructure Management および Assisted Installer 機能を使用します。Hosted control plane のホステッドクラスターは、HyperShift Operator を使用してプロビジョニングされます。

- クラスター作成時の追加のマニフェストの設定
- Amazon Web Services でのクラスターの作成
- Amazon Web Services GovCloud でのクラスターの作成

- Microsoft Azure でのクラスターの作成
- Google Cloud Platform でのクラスターの作成
- VMware vSphere でのクラスターの作成
- Red Hat OpenStack Platform でのクラスターの作成
- Red Hat Virtualization でのクラスターの作成 (非推奨)
- オンプレミス環境でのクラスターの作成
- Hosted Control Plane

1.5.4.1. CLI を使用したクラスターの作成

Kubernetes Operator 用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

1.5.4.1.1. 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。ただし、**2.x** は 2.5 に置き換えてください。

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/backplane-<2.x>
```

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。

注記: Nutanix プラットフォームを使用する場合は、**ClusterImageSet** リソースの **releaseImage** に **x86_64** アーキテクチャーを使用し、**visible** のラベル値を **'true'** に設定してください。以下の例を参照してください。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: stable
    visible: 'true'
```

```
  name: img4.x.47-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.x.47-x86_64
```

1.5.4.1.2. ClusterDeployment を使用してクラスターを作成する

ClusterDeployment は、クラスターのライフサイクルを制御するために使用される Hive カスタムリソースです。

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、個別のクラスターを作成します。

1.5.4.1.3. 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。

Cluster Pools のドキュメントに従って、Hive **ClusterPool** API でクラスターを作成します。

1.5.4.2. クラスター作成時の追加のマニフェストの設定

追加の Kubernetes リソースマニフェストは、クラスター作成のインストールプロセス中に設定できます。これは、ネットワークの設定やロードバランサーの設定など、シナリオの追加マニフェストを設定する必要がある場合に役立ちます。

クラスターを作成する前に、追加のリソースマニフェストが含まれる **ConfigMap** を指定する **ClusterDeployment** リソースへの参照を追加する必要があります。

注記: ClusterDeployment リソースと **ConfigMap** は同じ namespace にある必要があります。以下の例で、どのような内容かを紹介しています。

- リソースマニフェストを含む ConfigMap
  ConfigMap リソースが別のマニフェストが含まれる **ConfigMap**。リソースマニフェストの **ConfigMap** には、**data.<resource_name>\.yaml** パターンに追加されたリソース設定が指定されたキーを複数含めることができます。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: <my-baremetal-cluster-install-manifests>
  namespace: <mynamespace>
data:
  99_metal3-config.yaml: |
    kind: ConfigMap
    apiVersion: v1
    metadata:
      name: metal3-config
      namespace: openshift-machine-api
    data:
      http_port: "6180"
      provisioning_interface: "enp1s0"
      provisioning_ip: "172.00.0.3/24"
      dhcp_range: "172.00.0.10,172.00.0.100"
      deploy_kernel_url: "http://172.00.0.3:6180/images/ironic-python-agent.kernel"
      deploy_ramdisk_url: "http://172.00.0.3:6180/images/ironic-python-agent.initramfs"
```

```
      ironic_endpoint: "http://172.00.0.3:6385/v1/"
      ironic_inspector_endpoint: "http://172.00.0.3:5150/v1/"
      cache_url: "http://192.168.111.1/images"
      rhcos_image_url: "https://releases-art-
rhcos.svc.ci.openshift.org/art/storage/releases/rhcos-
4.3/43.81.201911192044.0/x86_64/rhcos-43.81.201911192044.0-
openstack.x86_64.qcow2.gz"
```

- リソースマニフェスト **ConfigMap** が参照される ClusterDeployment
  リソースマニフェスト **ConfigMap** は **spec.provisioning.manifestsConfigMapRef** で参照されます。

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: <my-baremetal-cluster>
  namespace: <mynamespace>
  annotations:
    hive.openshift.io/try-install-once: "true"
spec:
  baseDomain: test.example.com
  clusterName: <my-baremetal-cluster>
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    baremetal:
      libvirtSSHPrivateKeySecretRef:
        name: provisioning-host-ssh-private-key
  provisioning:
    installConfigSecretRef:
      name: <my-baremetal-cluster-install-config>
    sshPrivateKeySecretRef:
      name: <my-baremetal-hosts-ssh-private-key>
    manifestsConfigMapRef:
      name: <my-baremetal-cluster-install-manifests>
    imageSetRef:
      name: <my-clusterimageset>
    sshKnownHosts:
    - "10.1.8.90 ecdsa-sha2-nistp256
AAAAE2VjZHNhLXvVVVKUYVkuyvkuygkuyTCYTytfkufTYAAAAIbmlzdHAyNTYAAABBBKWjJR
zeUVuZs4yxSy4eu45xiANFIIbwE3e1aPzGD58x/NX7Yf+S8eFKq4RrsfSaK2hVJyJjvVIhUsU9z2s
BJP8="
  pullSecretRef:
    name: <my-baremetal-cluster-pull-secret>
```

1.5.4.3. Amazon Web Services でのクラスターの作成

multicluster engine Operator コンソールを使用して、Amazon Web Services (AWS)に Red Hat OpenShift Container Platform クラスターを作成できます。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの AWS へのインストール でプロセスの詳細を確認してください。

- 前提条件
- AWS クラスターの作成
- コンソールを使用したクラスターの作成

1.5.4.3.1. 前提条件

AWS でクラスターを作成する前に、以下の前提条件を確認してください。

- ハブクラスターをデプロイしている。
- AWS 認証情報がある。詳細は、Amazon Web Services の認証情報の作成 を参照してください。
- AWS で設定されたドメインがある。ドメインの設定方法は、AWS アカウントの設定 を参照してください。
- ユーザー名、パスワード、アクセスキー ID およびシークレットアクセスキーなど、Amazon Web Services (AWS) のログイン認証情報がある。Understanding and Getting Your Security Credentials を参照してください。
- OpenShift Container Platform イメージのプルシークレットがある。イメージプルシークレットの使用 を参照してください。
  注記: クラウドプロバイダーでクラウドプロバイダーのアクセスキーを変更する場合は、コンソールでクラウドプロバイダーの対応する認証情報を手動で更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

1.5.4.3.2. AWS クラスターの作成

AWS クラスターの作成に関する次の重要な情報を参照してください。

- クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML: On** を選択して、パネルに **install-config.yaml** ファイルの内容を表示できます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。
- クラスターを作成すると、コントローラーはクラスターとリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。
- クラスターを**破棄**すると、namespace とその中のすべてのリソースが削除されます。
- クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。
- 指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。
- マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

- AWS アカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更できます。この名前はクラスターのホスト名で使用されます。

- このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。利用可能なイメージのリストからイメージを選択します。使用したいイメージがない場合は、使用したいイメージの URL を入力します。

- ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。情報には以下のフィールドが含まれます。

  - region: ノードプールが必要なリージョンを指定します。

  - CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

  - ゾーン: コントロールプレーンプールを実行する場所を指定します。より分散されているコントロールプレーンノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。

  - インスタンスタイプ: コントロールプレーンノードのインスタンスタイプを指定します。インスタンスの作成後にインスタンスのタイプやサイズを変更できます。

  - ルートストレージ: クラスターに割り当てるルートストレージの量を指定します。

- ワーカープールにワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。オプションの情報には以下のフィールドが含まれます。

  - ゾーン: ワーカープールを実行する場所を指定します。より分散されているノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。

  - インスタンスタイプ: ワーカープールのインスタンスタイプを指定します。インスタンスの作成後にインスタンスのタイプやサイズを変更できます。

  - ノード数: ワーカープールのノード数を指定します。ワーカープールを定義する場合にこの設定は必須です。

  - ルートストレージ: ワーカープールに割り当てるルートストレージの量を指定します。ワーカープールを定義する場合にこの設定は必須です。

- クラスターにはネットワークの詳細が必要であり、IPv6 を使用するには複数のネットワークが必要です。**Add network** をクリックして、追加のネットワークを追加できます。

- 認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

  - HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL を指定します。

- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL を指定します。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
- プロキシーサイトなし: プロキシーをバイパスする必要のあるサイトのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

1.5.4.3.3. コンソールを使用したクラスターの作成

新しいクラスターを作成するには、次の手順を参照してください。代わりに既存のクラスターを **インポート** する場合は、クラスターのインポート を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

1. **Infrastructure** > **Clusters** に移動します。
2. **Clusters** ページで、以下を実行します。**Cluster** > **Create cluster** をクリックし、コンソールで手順を完了します。
3. **任意:** コンソールに情報を入力するときにコンテンツの更新を表示するには、**YAML: On** を選択します。

認証情報を作成する必要がある場合は、Amazon Web Services の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

1.5.4.3.4. 関連情報

- AWS プライベート設定情報は、AWS GovCloud クラスターの作成時に使用されます。その環境での クラスターの作成は、Amazon Web Services GovCloud でのクラスターの作成 を参照してください。
- 詳細は、AWS アカウントの設定 を参照してください。
- リリースイメージの詳細については、リリースイメージ を参照してください。
- サポートされているインスタントタイプの詳細は、AWS 汎用インスタンス などのクラウドプロバイダーのサイトにアクセスしてください。

1.5.4.4. Amazon Web Services GovCloud でのクラスターの作成

コンソールを使用して、Amazon Web Services (AWS) または AWS GovCloud で Red Hat OpenShift Container Platform クラスターを作成できます。この手順では、AWS GovCloud でクラスターを作成する方法を説明します。AWS でクラスターを作成する手順については、Amazon Web Services でのクラスターの作成 を参照してください。

AWS GovCloud は、政府のドキュメントをクラウドに保存するために必要な追加の要件を満たすクラウドサービスを提供します。AWS GovCloud でクラスターを作成する場合、環境を準備するために追加の手順を実行する必要があります。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について質問がある場合は、プロセスの詳細について、OpenShift Container Platform ドキュメントの AWS 上のクラスターを政府リージョンにインストールする を参照してください。以下のセクションでは、AWS GovCloud でクラスターを作成する手順を説明します。

- 前提条件
- Hive を AWS GovCloud にデプロイするように設定します。
- コンソールを使用したクラスターの作成

1.5.4.4.1. 前提条件

AWS GovCloud クラスターを作成する前に、以下の前提条件を満たす必要があります。

- ユーザー名、パスワード、アクセスキー ID、およびシークレットアクセスキーなどの AWS ログイン認証情報が必要です。Understanding and Getting Your Security Credentials を参照してください。
- AWS 認証情報がある。詳細は、Amazon Web Services の認証情報の作成 を参照してください。
- AWS で設定されたドメインがある。ドメインの設定方法は、AWS アカウントの設定 を参照してください。
- OpenShift Container Platform イメージのプルシークレットがある。イメージプルシークレットの使用 を参照してください。
- ハブクラスター用の既存の Red Hat OpenShift Container Platform クラスターを備えた Amazon Virtual Private Cloud (VPC) が必要です。この VPC は、マネージドクラスターリソースまたはマネージドクラスターサービスエンドポイントに使用される VPC とは異なる必要があります。
- マネージドクラスターリソースがデプロイされる VPC が必要です。これは、ハブクラスターまたはマネージドクラスターサービスエンドポイントに使用される VPC と同じにすることはできません。
- マネージドクラスターサービスエンドポイントを提供する 1 つ以上の VPC が必要です。これは、ハブクラスターまたはマネージドクラスターリソースに使用される VPC と同じにすることはできません。
- Classless Inter-Domain Routing (CIDR) によって指定される VPC の IP アドレスが重複しないようにしてください。
- Hive namespace 内で認証情報を参照する **HiveConfig** カスタムリソースが必要です。このカスタムリソースは、マネージドクラスターサービスエンドポイント用に作成した VPC でリソースを作成するためにアクセスできる必要があります。

注記: クラウドプロバイダーでクラウドプロバイダーのアクセスキーを変更する場合は、multicluster engine Operator コンソールでクラウドプロバイダーの対応する認証情報を手動で更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

1.5.4.4.2. Hive を AWS GovCloud にデプロイするように設定します。

AWS GovCloud でのクラスターの作成は、標準の AWS でクラスターを作成することとほぼ同じですが、AWS GovCloud でクラスターの AWS PrivateLink を準備するために追加の手順を実行する必要があります。

1.5.4.4.2.1. リソースおよびエンドポイントの VPC の作成

前提条件に記載されているように、ハブクラスターが含まれる VPC に加えて、2 つの VPC が必要です。VPC を作成する具体的な手順については、Amazon Web Services ドキュメントの VPC の作成 を参照してください。

1. プライベートサブネットを使用してマネージドクラスターの VPC を作成します。
2. プライベートサブネットを使用して、マネージドクラスターサービスエンドポイントの 1 つ以上の VPC を作成します。リージョンの各 VPC には 255 VPC エンドポイントの制限があるため、そのリージョン内の 255 を超えるクラスターをサポートするには、複数の VPC が必要です。
3. 各 VPC について、リージョンのサポートされるすべてのアベイラビリティーゾーンにサブネットを作成します。コントローラーの要件があるため、各サブネットには少なくとも 255 以上の使用可能な IP アドレスが必要です。
   以下の例は、**us-gov-east-1** リージョンに 6 つのアベイラビリティーゾーンを持つ VPC のサブネットを設定する方法を示しています。

   ```
   vpc-1 (us-gov-east-1) : 10.0.0.0/20
     subnet-11 (us-gov-east-1a): 10.0.0.0/23
     subnet-12 (us-gov-east-1b): 10.0.2.0/23
     subnet-13 (us-gov-east-1c): 10.0.4.0/23
     subnet-12 (us-gov-east-1d): 10.0.8.0/23
     subnet-12 (us-gov-east-1e): 10.0.10.0/23
     subnet-12 (us-gov-east-1f): 10.0.12.0/2
   ```

   ```
   vpc-2 (us-gov-east-1) : 10.0.16.0/20
     subnet-21 (us-gov-east-1a): 10.0.16.0/23
     subnet-22 (us-gov-east-1b): 10.0.18.0/23
     subnet-23 (us-gov-east-1c): 10.0.20.0/23
     subnet-24 (us-gov-east-1d): 10.0.22.0/23
     subnet-25 (us-gov-east-1e): 10.0.24.0/23
     subnet-26 (us-gov-east-1f): 10.0.28.0/23
   ```

4. すべてのハブクラスター (ハブクラスター VPC) に、ピアリング、転送ゲートウェイ、およびすべての DNS 設定が有効になっている VPC エンドポイント用に作成した VPC へのネットワーク接続があることを確認します。
5. AWS GovCloud 接続に必要な AWS PrivateLink の DNS 設定を解決するために必要な VPC の一覧を収集します。これには、設定している multicluster engine Operator インスタンスの VPC が少なくとも含まれ、さまざまな Hive コントローラーが存在するすべての VPC の一覧を含めることができます。

1.5.4.4.2.2. VPC エンドポイントのセキュリティーグループの設定

AWS の各 VPC エンドポイントには、エンドポイントへのアクセスを制御するためにセキュリティーグループが割り当てられます。Hive が VPC エンドポイントを作成する場合、セキュリティーグループは指定しません。VPC のデフォルトのセキュリティーグループは VPC エンドポイントに割り当てられま

す。VPC のデフォルトのセキュリティーグループには、VPC エンドポイントが Hive インストーラー Pod から作成されるトラフィックを許可するルールが必要です。詳細については、AWS ドキュメントの エンドポイントポリシーを使用した VPC エンドポイントへのアクセスの制御 を参照してください。

たとえば、Hive が **hive-vpc (10.1.0.0/16)** で実行されている場合は、VPC エンドポイントが作成される VPC のデフォルトセキュリティーグループに、**10.1.0.0/16** からのイングレスを許可するルールが必要です。

1.5.4.4.2.3. AWS PrivateLink の権限の設定

AWS PrivateLink を設定するには、複数の認証情報が必要です。これらの認証情報に必要な権限は、認証情報のタイプによって異なります。

- ClusterDeployment の認証情報には、以下の権限が必要です。

```
ec2:CreateVpcEndpointServiceConfiguration
ec2:DescribeVpcEndpointServiceConfigurations
ec2:ModifyVpcEndpointServiceConfiguration
ec2:DescribeVpcEndpointServicePermissions
ec2:ModifyVpcEndpointServicePermissions
ec2:DeleteVpcEndpointServiceConfigurations
```

- エンドポイント VPC アカウントの HiveConfig の認証情報 **.spec.awsPrivateLink.credentialsSecretRef** には、以下の権限が必要です。

```
ec2:DescribeVpcEndpointServices
ec2:DescribeVpcEndpoints
ec2:CreateVpcEndpoint
ec2:CreateTags
ec2:DescribeNetworkInterfaces
ec2:DescribeVPCs

ec2:DeleteVpcEndpoints

route53:CreateHostedZone
route53:GetHostedZone
route53:ListHostedZonesByVPC
route53:AssociateVPCWithHostedZone
route53:DisassociateVPCFromHostedZone
route53:CreateVPCAssociationAuthorization
route53:DeleteVPCAssociationAuthorization
route53:ListResourceRecordSets
route53:ChangeResourceRecordSets

route53:DeleteHostedZone
```

- VPC をプライベートホストゾーンに関連付けるために **HiveConfig** カスタムリソースに指定された認証情報 (**.spec.awsPrivateLink.associatedVPCs[$idx].credentialsSecretRef**)。VPC が置かれているアカウントには、以下の権限が必要です。

```
route53:AssociateVPCWithHostedZone
route53:DisassociateVPCFromHostedZone
ec2:DescribeVPCs
```

ハブクラスターの Hive namespace 内に認証情報シークレットがあることを確認します。

HiveConfig カスタムリソースは、特定の提供される VPC でリソースを作成する権限を持つ Hive namespace 内で認証情報を参照する必要があります。AWS GovCloud での AWS クラスターのプロビジョニングに使用する認証情報がすでに Hive namespace にある場合は、別の認証情報を作成する必要はありません。AWS GovCloud での AWS クラスターのプロビジョニングに使用する認証情報がまだ Hive namespace にない場合、現在の認証情報を置き換えるか、Hive namespace に追加の認証情報を作成できます。

HiveConfig カスタムリソースには、以下の内容が含まれている必要があります。

- 指定された VPC のリソースをプロビジョニングするために必要な権限を持つ AWS GovCloud 認証情報。
- OpenShift Container Platform クラスターインストールの VPC のアドレス、およびマネージドクラスターのサービスエンドポイント。
  ベストプラクティス: OpenShift Container Platform クラスターのインストールおよびサービスエンドポイントに異なる VPC を使用します。

以下の例は、認証情報の内容を示しています。

```
spec:
  awsPrivateLink:
    ## The list of inventory of VPCs that can be used to create VPC
    ## endpoints by the controller.
    endpointVPCInventory:
    - region: us-east-1
      vpcID: vpc-1
      subnets:
      - availabilityZone: us-east-1a
        subnetID: subnet-11
      - availabilityZone: us-east-1b
        subnetID: subnet-12
      - availabilityZone: us-east-1c
        subnetID: subnet-13
      - availabilityZone: us-east-1d
        subnetID: subnet-14
      - availabilityZone: us-east-1e
        subnetID: subnet-15
      - availabilityZone: us-east-1f
        subnetID: subnet-16
    - region: us-east-1
      vpcID: vpc-2
      subnets:
      - availabilityZone: us-east-1a
        subnetID: subnet-21
      - availabilityZone: us-east-1b
        subnetID: subnet-22
      - availabilityZone: us-east-1c
        subnetID: subnet-23
      - availabilityZone: us-east-1d
        subnetID: subnet-24
      - availabilityZone: us-east-1e
        subnetID: subnet-25
      - availabilityZone: us-east-1f
        subnetID: subnet-26
    ## The credentialsSecretRef points to a secret with permissions to create.
    ## The resources in the account where the inventory of VPCs exist.
```

```
  credentialsSecretRef:
    name: <hub-account-credentials-secret-name>

  ## A list of VPC where various mce clusters exists.
  associatedVPCs:
  - region: region-mce1
    vpcID: vpc-mce1
    credentialsSecretRef:
      name: <credentials-that-have-access-to-account-where-MCE1-VPC-exists>
  - region: region-mce2
    vpcID: vpc-mce2
    credentialsSecretRef:
      name: <credentials-that-have-access-to-account-where-MCE2-VPC-exists>
```

AWS PrivateLink が **endpointVPCInventory** 一覧でサポートされているすべてのリージョンから VPC を含めることができます。コントローラーは、ClusterDeployment の要件を満たす VPC を選択します。

詳細は、Hive ドキュメント を参照してください。

1.5.4.4.3. コンソールを使用したクラスターの作成

コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** > **Create cluster AWS** > **Standalone** に移動して、コンソールで手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

AWS GovCloud クラスターを作成する場合、選択する認証情報は AWS GovCloud リージョンのリソースにアクセスできる必要があります。クラスターをデプロイするために必要な権限を持つ場合は、Hive namespace にある AWS GovCloud シークレットを使用できます。コンソールに既存の認証情報が表示されます。認証情報を作成する必要がある場合は、Amazon Web Services の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: YAML: On を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

AWS または AWS GovCloud アカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更

できます。この名前はクラスターのホスト名で使用されます。詳細は、AWS アカウントの設定 を参照してください。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。

ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。情報には以下のフィールドが含まれます。

- リージョン: クラスターリソースを作成するリージョン。AWS GovCloud プロバイダーでクラスターを作成する場合、ノードプールの AWS GovCloud リージョンを含める必要があります。たとえば、**us-gov-west-1** です。
- CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。
- ゾーン: コントロールプレーンプールを実行する場所を指定します。より分散されているコントロールプレーンノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。
- インスタンスタイプ: コントロールプレーンノードのインスタンスタイプを指定します。これは、以前に指定した **CPU アーキテクチャー** と同じにする必要があります。インスタンスの作成後にインスタンスのタイプやサイズを変更できます。
- ルートストレージ: クラスターに割り当てるルートストレージの量を指定します。

ワーカープールにワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。オプションの情報には以下のフィールドが含まれます。

- プール名: プールの一意の名前を指定します。
- ゾーン: ワーカープールを実行する場所を指定します。より分散されているノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。
- インスタンスタイプ: ワーカープールのインスタンスタイプを指定します。インスタンスの作成後にインスタンスのタイプやサイズを変更できます。
- ノード数: ワーカープールのノード数を指定します。ワーカープールを定義する場合にこの設定は必須です。
- ルートストレージ: ワーカープールに割り当てるルートストレージの量を指定します。ワーカープールを定義する場合にこの設定は必須です。

クラスターにはネットワークの詳細が必要であり、IPv6 を使用するには複数のネットワークが必要です。AWS GovCloud クラスターの場合は、**Machine CIDR** フィールドに Hive VPC のアドレスのブロックの値を入力します。**Add network** をクリックして、追加のネットワークを追加できます。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー URL: **HTTP** トラフィックのプロキシーとして使用する URL を指定します。
- HTTPS プロキシー URL: **HTTPS** トラフィックに使用するセキュアなプロキシー URL を指定します。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
- プロキシードメインなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

AWS GovCloud クラスターを作成するか、プライベート環境を使用する場合は、AMI ID およびサブネット値を使用して、**AWS プライベート設定** ページのフィールドに入力します。**ClusterDeployment.yaml** ファイルで **spec:platform:aws:privateLink:enabled** の値が **true** に設定されていることを確認します。これは、**Use private configuration** を選択すると自動的に設定されます。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML: On** を選択して、パネルに **install-config.yaml** ファイルの内容を表示できます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、Kubernetes Operator のマルチクラスターエンジンの管理下に自動的に設定されます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.5. Microsoft Azure でのクラスターの作成

multicluster engine Operator コンソールを使用して、Microsoft Azure または Microsoft Azure Government に Red Hat OpenShift Container Platform クラスターをデプロイできます。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの Azure へのインストール でプロセスの詳細を確認してください。

- 前提条件
- コンソールを使用したクラスターの作成

1.5.4.5.1. 前提条件

Azure でクラスターを作成する前に、以下の前提条件を確認してください。

- ハブクラスターをデプロイしている。
- Azure 認証情報がある。詳細は、Microsoft Azure の認証情報の作成 を参照してください。

- Azure または Azure Government に設定済みドメインがある。ドメイン設定の方法は、Configuring a custom domain name for an Azure cloud service を参照してください。
- ユーザー名とパスワードなどの Azure ログイン認証情報がある。Microsoft Azure Portal を参照してください。
- **clientId**、**clientSecret** および **tenantId** などの Azure サービスプリンシパルがある。azure.microsoft.com を参照してください。
- OpenShift Container Platform イメージプルシークレットがある。イメージプルシークレットの使用 を参照してください。

注記: クラウドプロバイダーでクラウドプロバイダーのアクセスキーを変更する場合は、multicluster engine operator のコンソールでクラウドプロバイダーの対応する認証情報を手動で更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

1.5.4.5.2. コンソールを使用したクラスターの作成

multicluster engine Operator コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** に移動します。**Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

認証情報を作成する必要がある場合は、詳細について Microsoft Azure の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: YAML: On を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

Azure アカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更できます。詳細は、Configuring a custom domain name for an Azure cloud service を参照してください。この名前はクラスターのホスト名で使用されます。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。

ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。この情報には、次のオプションフィールドが含まれます。

- リージョン: ノードプールを実行するリージョンを指定します。より分散されているコントロールプレーンノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。

- CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

クラスターの作成後に、コントロールプレーンプールのタイプおよびルートストレージの割り当て (必須) を変更できます。

ワーカープールに 1 つまたは複数のワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。情報には以下のフィールドが含まれます。

- ゾーン: ワーカープールを実行することを指定します。より分散されているノードグループでは、リージョンで複数のゾーンを選択できます。ゾーンが近くにある場合はパフォーマンスの速度が向上しますが、ゾーンの距離が離れると、より分散されます。

- インスタンスタイプ: インスタンスのタイプとサイズは、作成後に変更できます。

Add network をクリックして、追加のネットワークを追加できます。IPv6 アドレスを使用している場合は、複数のネットワークが必要です。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL。

- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。

- プロキシーなし: プロキシーをバイパスする必要のあるドメインのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。

- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML** スイッチをクリックして **On** にすると、パネルに **install-config.yaml** ファイルの内容が表示されます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.6. Google Cloud Platform でのクラスターの作成

Google Cloud Platform (GCP) で Red Hat OpenShift Container Platform クラスターを作成する手順に従います。GCP の詳細については、Google Cloud Platform を参照してください。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの GCP のインストール でプロセスの詳細を確認してください。

- 前提条件
- コンソールを使用したクラスターの作成

1.5.4.6.1. 前提条件

GCP でクラスターを作成する前に、次の前提条件を確認してください。

- ハブクラスターをデプロイしている。
- GCP 認証情報がある。詳細は、Google Cloud Platform の認証情報の作成 を参照してください。
- GCP に設定済みのドメインがある。ドメインの設定方法は、Setting up a custom domain を参照してください。
- ユーザー名とパスワードを含む GCP ログイン認証情報がある。
- OpenShift Container Platform イメージのプルシークレットがある。イメージプルシークレットの使用 を参照してください。

注記: クラウドプロバイダーでクラウドプロバイダーのアクセスキーを変更する場合は、multicluster engine operator のコンソールでクラウドプロバイダーの対応する認証情報を手動で更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

1.5.4.6.2. コンソールを使用したクラスターの作成

multicluster engine Operator コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** に移動します。**Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

認証情報を作成する必要がある場合は、Google Cloud Platform の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。GCP クラスターの命名に適用される制限がいくつかあります。この制限には、名前を **goog** で開始しないことや、名前に **google** に類似する文字および数字のグループが含まれないことなどがあります。制限の完全な一覧は、Bucket naming guidelines を参照してください。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: YAML: On を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

選択した GCP アカウントの認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更できます。詳細は、Setting up a custom domain を参照してください。この名前はクラスターのホスト名で使用されます。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。

ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。情報には以下のフィールドが含まれます。

- リージョン: コントロールプレーンプールを実行するリージョンを指定します。リージョンが近くにある場合はパフォーマンスの速度が向上しますが、リージョンの距離が離れると、より分散されます。
- CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

コントロールプレーンプールのインスタンスタイプを指定できます。インスタンスの作成後にインスタンスのタイプやサイズを変更できます。

ワーカープールに 1 つまたは複数のワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。情報には以下のフィールドが含まれます。

- インスタンスタイプ: インスタンスのタイプとサイズは、作成後に変更できます。
- ノード数: ワーカープールを定義するときに必要な設定です。

ネットワークの詳細が必要であり、IPv6 アドレスを使用するには複数のネットワークが必要です。**Add network** をクリックして、追加のネットワークを追加できます。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL。
- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
- プロキシーサイトなし: プロキシーをバイパスする必要のあるサイトのコンマ区切りリスト。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML: On** を選択して、パネルに **install-config.yaml** ファイルの内容を表示できます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.7. VMware vSphere でのクラスターの作成

multicluster engine Operator コンソールを使用して、VMware vSphere に Red Hat OpenShift Container Platform クラスターをデプロイできます。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの vSphere のインストール でプロセスの詳細を確認してください。

- 前提条件
- コンソールを使用したクラスターの作成

1.5.4.7.1. 前提条件

vSphere でクラスターを作成する前に、次の前提条件を確認してください。

- OpenShift Container Platform バージョン 4.13 以降にデプロイされたハブクラスターがある。
- vSphere 認証情報がある。詳細は、VMware vSphere の認証情報の作成 を参照してください。
- OpenShift Container Platform イメージプルシークレットがある。イメージプルシークレットの使用 を参照してください。
- デプロイする VMware インスタンスについて、以下の情報がある。
  - API および Ingress インスタンスに必要な静的 IP アドレス
  - 以下の DNS レコード。
    - 次の API ベースドメインは静的 API VIP を指す必要があります。

```
api.<cluster_name>.<base_domain>
```

- 次のアプリケーションベースドメインは、Ingress VIP の静的 IP アドレスを指す必要があります。

```
*.apps.<cluster_name>.<base_domain>
```

1.5.4.7.2. コンソールを使用したクラスターの作成

multicluster engine Operator コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** に移動します。**Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

認証情報を作成する必要がある場合は、認証情報の作成の詳細について、VMware vSphere の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: YAML: On を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

vSphere アカウントで設定した、選択した認証情報に関連付けられているベースドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更できます。詳細は、カスタマイズによる vSphere へのクラスターのインストール を参照してください。値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。この名前はクラスターのホスト名で使用されます。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。

注記: OpenShift Container Platform バージョン 4.13 以降のリリースイメージがサポートされています。

ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。この情報には、**CPU アーキテクチャー** フィールドが含まれます。次のフィールドの説明を表示します。

- CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

ワーカープールに 1 つまたは複数のワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。この情報には、**ソケットあたりのコア数**、CPU、Memory_min MiB、GiB **単位の** _Disk **サイズ**、および **ノード数** が含まれます。

ネットワーク情報が必要です。IPv6 を使用するには、複数のネットワークが必要です。必要なネットワーク情報の一部は、次のフィールドに含まれています。

- vSphere ネットワーク名: VMware vSphere ネットワーク名を指定します。

- API VIP: 内部 API 通信に使用する IP アドレスを指定します。
  注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **api.** が正しく解決されるようにします。

- Ingress VIP: Ingress トラフィックに使用する IP アドレスを指定します。
  注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **test.apps.** が正しく解決されるようにします。

Add network をクリックして、追加のネットワークを追加できます。IPv6 アドレスを使用している場合は、複数のネットワークが必要です。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL を指定します。

- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL を指定します。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。

- プロキシーサイトなし: プロキシーをバイパスする必要のあるサイトのコンマ区切りリストを指定します。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。

- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

Disconnected installation をクリックして、オフラインインストールイメージを定義できます。Red Hat OpenStack Platform プロバイダーとオフラインインストールを使用してクラスターを作成する際に、ミラーレジストリーにアクセスするための証明書が必要な場合は、クラスターを作成する際の認証情報または **Disconnected installation** セクションを設定するときに、**Configuration for disconnected installation** セクションの **Additional trust bundle** フィールドにその証明書を入力する必要があります。

Add automation template をクリックしてテンプレートを作成できます。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML** スイッチをクリックして **On** にすると、パネルに **install-config.yaml** ファイルの内容が表示されます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.8. Red Hat OpenStack Platform でのクラスターの作成

multicluster engine Operator コンソールを使用して、Red Hat OpenStack Platform に Red Hat OpenShift Container Platform クラスターをデプロイできます。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの OpenStack のインストール でプロセスの詳細を確認してください。

- 前提条件
- コンソールを使用したクラスターの作成

1.5.4.8.1. 前提条件

Red Hat OpenStack Platform でクラスターを作成する前に、以下の前提条件を確認してください。

- OpenShift Container Platform バージョン 4.6 以降にデプロイされたハブクラスターが必要です。
- Red Hat OpenStack Platform の認証情報がある。詳細は、Red Hat OpenStack Platform の認証情報の作成 を参照してください。
- OpenShift Container Platform イメージプルシークレットがある。イメージプルシークレットの使用 を参照してください。
- デプロイする Red Hat OpenStack Platform インスタンスについての以下の情報がある。
  - コントロールプレーンとワーカーインスタンスのフレーバー名 (例:**m1.xlarge**)
  - Floating IP アドレスを提供する外部ネットワークのネットワーク名
  - API および Ingress インスタンスに必要な静的 IP アドレス
  - 以下の DNS レコード。
    - 次の API ベースドメインは、API のフローティング IP アドレスを指す必要があります。

      ```
      api.<cluster_name>.<base_domain>
      ```

    - 次のアプリケーションベースドメインは、ingress:app-name のフローティング IP アドレスを指す必要があります。

      ```
      *.apps.<cluster_name>.<base_domain>
      ```

1.5.4.8.2. コンソールを使用したクラスターの作成

multicluster engine Operator コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** に移動します。**Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

認証情報を作成する必要がある場合は、Red Hat OpenStack Platform の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。名前には 15 文字以上指定できません。値は、認証情報の要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: **YAML: On** を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

Red Hat OpenStack Platform アカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は、上書きすると変更できます。詳細は、Red Hat OpenStack Platform ドキュメントの ドメインの管理 を参照してください。この名前はクラスターのホスト名で使用されます。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。OpenShift Container Platform バージョン 4.6.x 以降のリリースイメージのみがサポートされます。

ノードプールには、コントロールプレーンプールとワーカープールが含まれます。コントロールプレーンノードは、クラスターアクティビティーの管理を共有します。マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

コントロールプレーンプールのインスタンスタイプを追加する必要がありますが、インスタンスの作成後にインスタンスのタイプとサイズを変更できます。

ワーカープールに 1 つまたは複数のワーカーノードを作成し、クラスターのコンテナーワークロードを実行できます。ワーカーノードは、1 つのワーカープールに属することも、複数のワーカープールに分散させることもできます。ワーカーノードが指定されていない場合は、コントロールプレーンノードもワーカーノードとして機能します。情報には以下のフィールドが含まれます。

- インスタンスタイプ: インスタンスのタイプとサイズは、作成後に変更できます。
- ノード数: ワーカープールのノード数を指定します。ワーカープールを定義する場合にこの設定は必須です。

クラスターにはネットワークの詳細が必要です。IPv4 ネットワーク用に 1 つ以上のネットワークの値を指定する必要があります。IPv6 ネットワークの場合は、複数のネットワークを定義する必要があります。

Add network をクリックして、追加のネットワークを追加できます。IPv6 アドレスを使用している場合は、複数のネットワークが必要です。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL を指定します。
- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL。値が指定されていない場合、**HTTP Proxy** と同じ値が **HTTP** と **HTTPS** の両方に使用されます。
- プロキシーなし: プロキシーをバイパスする必要のあるサイトのコンマ区切りリストを定義します。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

Disconnected installation をクリックして、オフラインインストールイメージを定義できます。Red Hat OpenStack Platform プロバイダーとオフラインインストールを使用してクラスターを作成する際に、ミラーレジストリーにアクセスするための証明書が必要な場合は、クラスターを作成する際の認証情報または **Disconnected installation** セクションを設定するときに、**Configuration for disconnected installation** セクションの **Additional trust bundle** フィールドにその証明書を入力する必要があります。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML** スイッチをクリックして **On** にすると、パネルに **install-config.yaml** ファイルの内容が表示されます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

内部認証局 (CA) を使用するクラスターを作成する場合、以下の手順を実行してクラスターの YAML ファイルをカスタマイズする必要があります。

1. レビューステップで **YAML** スイッチをオンにし、CA 証明書バンドルを使用してリストの上部に **Secret** オブジェクトを挿入します。**注記:** Red Hat OpenStack Platform 環境が複数の機関によって署名された証明書を使用してサービスを提供する場合、バンドルには、必要なすべてのエンドポイントを検証するための証明書を含める必要があります。**ocp3** という名前のクラスターの追加は以下の例のようになります。

```
apiVersion: v1
kind: Secret
type: Opaque
metadata:
  name: ocp3-openstack-trust
  namespace: ocp3
stringData:
  ca.crt: |
    -----BEGIN CERTIFICATE-----
    <Base64 certificate contents here>
```

```
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Base64 certificate contents here>
-----END CERTIFICATE----
```

2. 以下の例のように、Hive **ClusterDeployment** オブジェクトを変更して、**spec.platform.openstack** に **certificatesSecretRef** の値を指定します。

```
platform:
  openstack:
    certificatesSecretRef:
      name: ocp3-openstack-trust
    credentialsSecretRef:
      name: ocp3-openstack-creds
    cloud: openstack
```

上記の例では、**clouds.yaml** ファイルのクラウド名が **openstack** であることを前提としています。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.9. Red Hat Virtualization でのクラスターの作成 (非推奨)

非推奨: Red Hat Virtualization 認証情報とクラスター作成機能は非推奨となり、サポートされなくなりました。

multicluster engine Operator コンソールを使用して、Red Hat Virtualization に Red Hat OpenShift Container Platform クラスターを作成できます。

クラスターを作成する場合、作成プロセスでは、Hive リソースを使用して OpenShift Container Platform インストーラーを使用します。この手順の完了後にクラスターの作成について不明な点がある場合は、OpenShift Container Platform ドキュメントの RHV へのインストール でプロセスの詳細を確認してください。

- 前提条件
- コンソールを使用したクラスターの作成

1.5.4.9.1. 前提条件

Red Hat Virtualization でクラスターを作成する前に、以下の前提条件を確認してください (非推奨)。

- ハブクラスターをデプロイしている。
- Red Hat Virtualization の認証情報がある。詳細は、Red Hat Virtualization の認証情報の作成 を参照してください。

- oVirt Engine 仮想マシンに設定されたドメインおよび仮想マシンプロキシーがある。ドメインの設定方法は、Red Hat OpenShift Container Platform ドキュメントの RHV へのインストール を参照してください。
- Red Hat Virtualization のログイン認証情報 (Red Hat カスタマーポータルのユーザー名およびパスワードを含む) がある。
- OpenShift Container Platform イメージプルシークレットがある。プルシークレットは、Pull secret ページからダウンロードできます。プルシークレットの詳細は、イメージプルシークレットの使用 を参照してください。

注記: クラウドプロバイダーでクラウドプロバイダーのアクセスキーを変更する場合は、multicluster engine operator のコンソールでクラウドプロバイダーの対応する認証情報を手動で更新する必要もあります。これは、マネージドクラスターがホストされ、マネージドクラスターの削除を試みるクラウドプロバイダーで認証情報の有効期限が切れる場合に必要です。

- 次の DNS レコードが必要です。
  - 次の API ベースドメインは静的 API VIP を指す必要があります。

    ```
    api.<cluster_name>.<base_domain>
    ```

  - 次のアプリケーションベースドメインは、Ingress VIP の静的 IP アドレスを指す必要があります。

    ```
    *.apps.<cluster_name>.<base_domain>
    ```

1.5.4.9.2. コンソールを使用したクラスターの作成

multicluster engine Operator コンソールからクラスターを作成するには、**Infrastructure** > **Clusters** に移動します。**Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。

注記: この手順では、クラスターを作成します。既存のクラスターをインポートする場合は、クラスターのインポート でインポート手順を参照してください。

認証情報を作成する必要がある場合は Red Hat Virtualization の認証情報の作成 を参照してください。

クラスターの名前はクラスターのホスト名で使用されます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

ヒント: **YAML: On** を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-admin** 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

Red Hat Virtualization アカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値を上書きして変更できます。

このリリースイメージで、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。リリースイメージの詳細については、リリースイメージ を参照してください。

コントロールプレーンプールのコア、ソケット、メモリー、およびディスクサイズの数など、ノードプールの情報。3 つのコントロールプレーンノードは、クラスターアクティビティーの管理を共有します。この情報には、**Architecture** フィールドが含まれます。次のフィールドの説明を表示します。

- CPU アーキテクチャー:マネージドクラスターのアーキテクチャータイプがハブクラスターのアーキテクチャーと同じでない場合は、プール内のマシンの命令セットアーキテクチャーの値を入力します。有効な値は **amd64**、**ppc64le**、**s390x**、および **arm64** です。

ワーカープール情報には、ワーカープールのプール名、コア数、メモリー割り当て、ディスクサイズ割り当て、およびノード数が必要です。ワーカープール内のワーカーノードは単一のワーカープールに配置するか、複数のワーカープールに分散できます。

事前設定された oVirt 環境には、以下のネットワークの詳細が必要です。

- oVirt ネットワーク名
- vNIC Profile ID: 仮想ネットワークインターフェイスカードのプロファイル ID を指定します。
- API VIP: 内部 API 通信に使用する IP アドレスを指定します。
  注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **api.** が正しく解決されるようにします。
- Ingress VIP: Ingress トラフィックに使用する IP アドレスを指定します。
  注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **test.apps.** が正しく解決されるようにします。
- ネットワークタイプ: デフォルト値は **OpenShiftSDN** です。IPv6 を使用するには、OVNKubernetes の設定は必須です。
- クラスターネットワーク CIDR: これは、Pod IP アドレスに使用できる IP アドレスの数およびリストです。このブロックは他のネットワークブロックと重複できません。デフォルト値は **10.128.0.0/14** です。
- ネットワークホストの接頭辞: 各ノードのサブネット接頭辞の長さを設定します。デフォルト値は **23** です。
- サービスネットワーク CIDR: サービスの IP アドレスのブロックを指定します。このブロックは他のネットワークブロックと重複できません。デフォルト値は **172.30.0.0/16** です。
- マシン CIDR: OpenShift Container Platform ホストで使用される IP アドレスのブロックを指定します。このブロックは他のネットワークブロックと重複できません。デフォルト値は **10.0.0.0/16** です。
  Add network をクリックして、追加のネットワークを追加できます。IPv6 アドレスを使用している場合は、複数のネットワークが必要です。

認証情報で提供されるプロキシー情報は、プロキシーフィールドに自動的に追加されます。情報をその

認証情報で提供されるプロキシー情報は、プロキシーのデフォルトに自動的に追加されます。情報をそのまま使用することも、上書きすることも、プロキシーを有効にする場合に情報を追加することもできます。次のリストには、プロキシーの作成に必要な情報が含まれています。

- HTTP プロキシー: **HTTP** トラフィックのプロキシーとして使用する URL を指定します。
- HTTPS プロキシー: **HTTPS** トラフィックに使用するセキュアなプロキシー URL を指定します。値の指定がない場合は、**HTTP Proxy URL** と同じ値が **HTTP** および **HTTPS** の両方に使用されます。
- プロキシーサイトなし: プロキシーをバイパスする必要のあるサイトのコンマ区切りリストを指定します。ドメイン名をピリオド (.) で開始し、そのドメインにあるすべてのサブドメインを組み込みます。アステリスク (*) を追加し、すべての宛先のプロキシーをバイパスします。
- 追加の信頼バンドル: HTTPS 接続のプロキシーに必要な 1 つ以上の追加の CA 証明書。

クラスターを作成する前に情報を確認し、必要に応じてカスタマイズする場合は、**YAML** スイッチをクリックして **On** にすると、パネルに **install-config.yaml** ファイルの内容が表示されます。更新がある場合は、カスタム設定で YAML ファイルを編集できます。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

注記: クラスターのインポートには、クラスターの詳細で提示された **oc** コマンドを実行する必要はありません。クラスターを作成すると、multicluster engine operator で管理されるように自動的に設定されます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.10. オンプレミス環境でのクラスターの作成

コンソールを使用して、オンプレミスの Red Hat OpenShift Container Platform クラスターを作成できます。クラスターに指定できるのは、VMware vSphere、Red Hat OpenStack、Red Hat Virtualization Platform (非推奨)、Nutanix、またはベアメタル環境上の、シングルノード OpenShift クラスター、マルチノードクラスター、およびコンパクトな 3 ノードクラスターです。

プラットフォームの値が **platform=none** に設定されているため、クラスターをインストールするプラットフォームとのプラットフォーム統合はありません。シングルノード OpenShift クラスターには、コントロールプレーンサービスとユーザーワークロードをホストするシングルノードのみが含まれます。この設定は、クラスターのリソースフットプリントを最小限に抑えたい場合に役立ちます。

Red Hat OpenShift Container Platform で利用できる機能であるゼロタッチプロビジョニング機能を使用して、エッジリソース上に複数のシングルノード OpenShift クラスターをプロビジョニングすることもできます。ゼロタッチプロビジョニングの詳細については、OpenShift Container Platform ドキュメントの **ネットワーク遠端のクラスター** を参照してください。

- 前提条件
- コンソールを使用したクラスターの作成
- コマンドラインを使用したクラスターの作成

1.5.4.10.1. 前提条件

オンプレミス環境にクラスターを作成する前に、以下の前提条件を確認してください。

- OpenShift Container Platform バージョン 4.13 以降にデプロイされたハブクラスターがある。
- 設定済みホストのホストインベントリーを備えた設定済みインフラストラクチャー環境がある。
- クラスターの作成に必要なイメージを取得できるように、ハブクラスターにインターネットアクセスがある (接続環境) か、インターネットに接続されている内部レジストリーまたはミラーレジストリーへの接続がある (非接続環境)。
- オンプレミス認証情報が設定されている。
- OpenShift Container Platform イメージプルシークレットがある。**イメージプルシークレットの使用** を参照してください。
- 次の DNS レコードが必要です。
  - 次の API ベースドメインは静的 API VIP を指す必要があります。

    ```
    api.<cluster_name>.<base_domain>
    ```

  - 次のアプリケーションベースドメインは、Ingress VIP の静的 IP アドレスを指す必要があります。

    ```
    *.apps.<cluster_name>.<base_domain>
    ```

1.5.4.10.2. コンソールを使用したクラスターの作成

コンソールからクラスターを作成するには、次の手順を実行します。

1. **Infrastructure** > **Clusters** に移動します。
2. **Clusters** ページで、**Create cluster** をクリックし、コンソールの手順を実行します。
3. クラスターのタイプとして **Host inventory** を選択します。

支援インストールでは、次のオプションを使用できます。

- **既存の検出されたホストを使用する**: 既存のホストインベントリーにあるホストのリストからホストを選択します。
- **新規ホストの検出**: 既存のインフラストラクチャー環境にないホストを検出します。インフラストラクチャー環境にあるものを使用するのではなく、独自のホストを検出します。

認証情報を作成する必要がある場合、詳細は **オンプレミス環境の認証情報の作成** を参照してください。

クラスターの名前は、クラスターのホスト名で使用されます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

注記: **YAML: On** を選択すると、コンソールに情報を入力する際にコンテンツの更新が表示されます。

クラスターを既存のクラスターセットに追加する場合は、そのクラスターセットで追加できる適切なパーミッションが必要です。クラスターの作成時に **cluster-admin** 権限がない場合に、**clusterset-**

admin 権限があるクラスターセットを選択する必要があります。指定されたクラスターセットに対して適切なパーミッションがないと、クラスターの作成に失敗します。選択するクラスターセットがない場合には、クラスター管理者に連絡して、任意のクラスターセットへの **clusterset-admin** 権限を受け取ってください。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、**デフォルト** のマネージドクラスターセットに自動的に追加されます。

プロバイダーアカウントで設定した、選択した認証情報に関連付けられているベース DNS ドメインがすでに存在する場合、その値がフィールドに入力されます。値は上書きすると変更できますが、この設定はクラスターの作成後には変更できません。プロバイダーのベースドメインは、Red Hat OpenShift Container Platform クラスターコンポーネントへのルートの作成に使用されます。これは、クラスタープロバイダーの DNS で Start of Authority (SOA) レコードとして設定されます。

OpenShift version は、クラスターの作成に使用される OpenShift Container Platform イメージのバージョンを特定します。使用するバージョンが利用可能な場合は、イメージの一覧からイメージを選択できます。標準イメージ以外のイメージを使用する場合は、使用するイメージの URL を入力できます。詳細は、**リリースイメージ** を参照してください。

サポート対象の OpenShift Container Platform バージョンを選択すると、**Install single node OpenShift** を選択するオプションが表示されます。シングルノード OpenShift クラスターには、コントロールプレーンサービスとユーザーワークロードをホストするシングルノードが含まれます。シングルノード OpenShift クラスターの作成後にノードを追加する方法の詳細は、**インフラストラクチャー環境へのホストのスケーリング** を参照してください。

クラスターをシングルノード OpenShift クラスターにする場合は、**シングルノード OpenShift** オプションを選択します。以下の手順を実行することで、シングルノードの OpenShift クラスターにワーカーを追加できます。

1. コンソールから、**Infrastructure** > **Clusters** に移動し、作成したクラスターまたはアクセスするクラスターの名前を選択します。
2. **Actions** > **Add hosts** を選択して、ワーカーを追加します。

注記: シングルノード OpenShift コントロールプレーンには 8 つの CPU コアが必要ですが、マルチノードコントロールプレーンクラスターのコントロールプレーンノードには 4 つの CPU コアしか必要ありません。

クラスターを確認して保存すると、クラスターはドラフトクラスターとして保存されます。**Clusters** ページでクラスター名を選択すると、作成プロセスを閉じてプロセスを終了することができます。

既存のホストを使用している場合は、ホストを独自に選択するか、自動的に選択するかどうかを選択します。ホストの数は、選択したノード数に基づいています。たとえば、シングルノード OpenShift クラスターではホストが 1 つだけ必要ですが、標準の 3 ノードクラスターには 3 つのホストが必要です。

このクラスターの要件を満たす利用可能なホストの場所は、**ホストの場所** のリストに表示されます。ホストと高可用性設定の分散については、複数の場所を選択します。

既存のインフラストラクチャー環境がない新しいホストを検出する場合は、**Discovery Image を使用したホストインベントリーへのホストの追加** の手順を実行します。

ホストがバインドされ、検証に合格したら、以下の IP アドレスを追加してクラスターのネットワーク情報を入力します。

- API VIP: 内部 API 通信に使用する IP アドレスを指定します。

注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させ

注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **api.** が正しく解決されるようにします。

- Ingress VIP: Ingress トラフィックに使用する IP アドレスを指定します。
  注記: 値は、要件セクションに記載されている DNS レコードの作成に使用した名前と一致させる必要があります。指定しない場合は、DNS を事前設定して **test.apps.** が正しく解決されるようにします。

Red Hat Advanced Cluster Management for Kubernetes を使用しており、マネージドクラスターの klusterlet を特定のノードで実行するように設定する場合は、必要な手順について オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

Clusters ナビゲーションページで、インストールのステータスを表示できます。

クラスターにアクセスする 手順については、クラスターへのアクセスに進みます。

1.5.4.10.3. コマンドラインを使用したクラスターの作成

Central Infrastructure Management 管理コンポーネント内のアシステッドインストーラー機能を使用して、コンソールを使用せずにクラスターを作成することもできます。この手順を完了したら、生成された検出イメージからホストを起動できます。通常、手順の順序は重要ではありませんが、順序が必要な場合は注意してください。

1.5.4.10.3.1. namespace を作成します。

リソースの namespace が必要です。すべてのリソースを共有 namespace に保持すると便利です。この例では、namespace の名前に **sample-namespace** を使用していますが、**assisted-installer** 以外の任意の名前を使用できます。次のファイルを作成して適用して namespace を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: sample-namespace
```

1.5.4.10.3.2. プルシークレットを namespace に追加する

以下のカスタムリソースを作成し、適用して プルシークレット を namespace に追加します。

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: <pull-secret>
  namespace: sample-namespace
stringData:
  .dockerconfigjson: 'your-pull-secret-json' 1
```

1 プルシークレットの内容を追加します。たとえば、これには **cloud.openshift.com**、**quay.io**、または **registry.redhat.io** 認証を含めることができます。

1.5.4.10.3.3. ClusterImageSet の生成

以下のカスタムリソースを作成して適用することで、**CustomImageSet** を生成してクラスターの OpenShift Container Platform のバージョンを指定します。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: openshift-v4.13.0
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.13.0-rc.0-x86_64
```

1.5.4.10.3.4. ClusterDeployment カスタムリソースを作成します。

ClusterDeployment カスタムリソース定義は、クラスターのライフサイクルを制御する API です。これは、クラスターリソースを定義する **spec.ClusterInstallRef** 設定で **AgentClusterInstall** カスタムリソースを参照します。

以下の例に基づいて **ClusterDeployment** カスタムリソースを作成して適用します。

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: single-node
  namespace: demo-worker4
spec:
  baseDomain: hive.example.com
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: test-agent-cluster-install (1)
    version: v1beta1
  clusterName: test-cluster
  controlPlaneConfig:
    servingCertificates: {}
  platform:
    agentBareMetal:
      agentSelector:
        matchLabels:
          location: internal
  pullSecretRef:
    name: <pull-secret> (2)
```

1. **AgentClusterInstall** リソースの名前を使用します。
2. Add the pull secret to the namespace でダウンロードしたプルシークレットを使用します。

1.5.4.10.3.5. AgentClusterInstall カスタムリソースを作成します。

AgentClusterInstall カスタムリソースでは、クラスターの要件の多くを指定できます。たとえば、クラスターネットワーク設定、プラットフォーム、コントロールプレーンの数、およびワーカーノードを指定できます。

次の例のようなカスタムリソースを作成して追加します。

```
apiVersion: extensions.hive.openshift.io/v1beta1
```

```
kind: AgentClusterInstall
metadata:
  name: test-agent-cluster-install
  namespace: demo-worker4
spec:
  platformType: BareMetal (1)
  clusterDeploymentRef:
    name: single-node (2)
  imageSetRef:
    name: openshift-v4.13.0 (3)
  networking:
    clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
    machineNetwork:
    - cidr: 192.168.111.0/24
    serviceNetwork:
    - 172.30.0.0/16
  provisionRequirements:
    controlPlaneAgents: 1
  sshPublicKey: ssh-rsa <your-public-key-here> (4)
```

1. クラスターが作成される環境のプラットフォームタイプを指定します。有効な値は、**BareMetal**、**None**、**VSphere**、**Nutanix**、または **External** です。
2. **ClusterDeployment** リソースに使用したものと同じ名前を使用します。
3. Generate a ClusterImageSet で生成した **ClusterImageSet** を使用します。
4. SSH 公開鍵を指定すると、インストール後にホストにアクセスできるようになります。

1.5.4.10.3.6. オプション: NMStateConfig カスタムリソースを作成する

NMStateConfig カスタムリソースは、静的 IP アドレスなどのホストレベルのネットワーク設定がある場合にのみ必要です。このカスタムリソースを含める場合は、**InfraEnv** カスタムリソースを作成する前にこの手順を完了する必要があります。**NMStateConfig** は、**InfraEnv** カスタムリソースの **spec.nmStateConfigLabelSelector** の値によって参照されます。

次の例のような **NMStateConfig** カスタムリソースを作成して適用します。必要に応じて値を置き換えます。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: NMStateConfig
metadata:
  name: <mynmstateconfig>
  namespace: <demo-worker4>
  labels:
    demo-nmstate-label: <value>
spec:
  config:
    interfaces:
      - name: eth0
        type: ethernet
        state: up
```

```
      mac-address: 02:00:00:80:12:14
      ipv4:
        enabled: true
        address:
          - ip: 192.168.111.30
            prefix-length: 24
        dhcp: false
    - name: eth1
      type: ethernet
      state: up
      mac-address: 02:00:00:80:12:15
      ipv4:
        enabled: true
        address:
          - ip: 192.168.140.30
            prefix-length: 24
        dhcp: false
  dns-resolver:
    config:
      server:
        - 192.168.126.1
  routes:
    config:
      - destination: 0.0.0.0/0
        next-hop-address: 192.168.111.1
        next-hop-interface: eth1
        table-id: 254
      - destination: 0.0.0.0/0
        next-hop-address: 192.168.140.1
        next-hop-interface: eth1
        table-id: 254
interfaces:
  - name: "eth0"
    macAddress: "02:00:00:80:12:14"
  - name: "eth1"
    macAddress: "02:00:00:80:12:15"
```

注記: demo-nmstate-label ラベル名と値は、**InfraEnv** リソースの **spec.nmStateConfigLabelSelector.matchLabels** フィールドに含める必要があります。

1.5.4.10.3.7. InfraEnv カスタムリソースを作成します。

InfraEnv カスタムリソースは、検出 ISO を作成する設定を提供します。このカスタムリソース内で、プロキシー設定、Ignition オーバーライドの値を特定し、**NMState** ラベルを指定します。このカスタムリソースの **spec.nmStateConfigLabelSelector** の値は、**NMStateConfig** カスタムリソースを参照します。

注: オプションの **NMStateConfig** カスタムリソースを含める場合は、**InfraEnv** カスタムリソースでそちらを参照する必要があります。**NMStateConfig** カスタムリソースを作成する前に **InfraEnv** カスタムリソースを作成した場合は、**InfraEnv** カスタムリソースを編集して **NMStateConfig** カスタムリソースを参照し、参照の追加後に ISO をダウンロードします。

以下のカスタムリソースを作成して適用します。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
```

```
metadata:
  name: myinfraenv
  namespace: demo-worker4
spec:
  clusterRef:
    name: single-node (1)
    namespace: demo-worker4 (2)
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <your_public_key_here>
  nmStateConfigLabelSelector:
    matchLabels:
      demo-nmstate-label: value
  proxy:
    httpProxy: http://USERNAME:PASSWORD@proxy.example.com:PORT
    httpsProxy: https://USERNAME:PASSWORD@proxy.example.com:PORT
    noProxy: .example.com,172.22.0.0/24,10.10.0.0/24
```

1. Create the ClusterDeployment の **clusterDeployment** リソース名を置き換えます。
2. ClusterDeployment の作成 の **clusterDeployment** リソース namespace を置き換えます。

1.5.4.10.3.7.1. InfraEnv のフィールドの表

フィールド	任意または必須	説明
sshAuthorizedKey	任意	SSH 公開鍵を指定できます。指定すると、検出 ISO イメージからホストを起動するときにホストにアクセスできるようになります。
nmStateConfigLabelSelector	任意	ホストの静的 IP、ブリッジ、ボンディングなどの高度なネットワーク設定を統合します。ホストネットワーク設定は、選択したラベルを持つ 1 つ以上の **NMStateConfig** リソースで指定されます。**nmStateConfigLabelSelector** プロパティーは、選択したラベルと一致する Kubernetes ラベルセレクターです。このラベルセレクターに一致するすべての **NMStateConfig** ラベルのネットワーク設定は、Discovery Image に含まれています。起動時に、各ホストは各設定をそのネットワークインターフェイスと比較し、適切な設定を適用します。

フィールド	任意または必須	説明
proxy	任意	proxy セクションでは、検出中にホストに必要なプロキシー設定を指定できます。

注記: IPv6 を使用してプロビジョニングする場合、**noProxy** 設定で CIDR アドレスブロックを定義することはできません。各アドレスを個別に定義する必要があります。

1.5.4.10.3.8. 検出イメージからホストを起動します。

残りの手順では、前の手順で取得した検出 ISO イメージからホストを起動する方法について説明します。

1. 次のコマンドを実行して、namespace から検出イメージをダウンロードします。

```
curl --insecure -o image.iso $(kubectl -n sample-namespace get infraenvs.agent-install.openshift.io myinfraenv -o=jsonpath="{.status.isoDownloadURL}")
```

2. 検出イメージを仮想メディア、USB ドライブ、または別の保管場所に移動し、ダウンロードしたディスカバリーイメージからホストを起動します。

3. **Agent** リソースは自動的に作成されます。これはクラスターに登録されており、検出イメージから起動したホストを表します。次のコマンドを実行して、**エージェントの** カスタムリソースを承認し、インストールを開始します。

```
oc -n sample-namespace patch agents.agent-install.openshift.io 07e80ea9-200c-4f82-aff4-4932acb773d4 -p '{"spec":{"approved":true}}' --type merge
```

エージェント名と UUID は、実際の値に置き換えます。

前のコマンドの出力に、**APPROVED** パラメーターの値が **true** であるターゲットクラスターのエントリーが含まれている場合、承認されたことを確認できます。

1.5.4.10.4. 関連情報

- CLI を使用して Nutanix プラットフォーム上にクラスターを作成するときに必要な追加の手順については、Red Hat OpenShift Container Platform ドキュメントの API を使用した Nutanix へのホストの追加 および Nutanix インストール後の設定 を参照してください。
- ゼロタッチプロビジョニングの詳細は、OpenShift Container Platform ドキュメントの ネットワーク遠端のクラスター を参照してください。
- イメージプルシークレットの使用 を参照してください。
- オンプレミス環境の認証情報の作成 を参照してください。
- リリースイメージ を参照してください。
- Discovery Image を使用したホストのホストインベントリーへの追加 を参照してください。

1.5.4.11. プロキシー環境でのクラスターの作成

ハブクラスターがプロキシーサーバー経由で接続されている場合は、Red Hat OpenShift Container Platform クラスターを作成できます。クラスターの作成を成功させるには、以下のいずれかの状況が true である必要があります。

- multicluster engine operator に、作成するマネージドクラスターとのプライベートネットワーク接続があり、マネージドクラスターがプロキシーを使用してインターネットにアクセスできる。
- マネージドクラスターはインフラストラクチャープロバイダーにあるが、ファイアウォールポートを使用することでマネージドクラスターからハブクラスターへの通信が可能になる。

プロキシーで設定されたクラスターを作成するには、以下の手順を実行します。

1. シークレットに保存されている **install-config** YAML に次の情報を追加して、ハブクラスターで **cluster-wide-proxy** 設定を設定します。

```
apiVersion: v1
kind: Proxy
baseDomain: <domain>
proxy:
  httpProxy: http://<username>:<password>@<proxy.example.com>:<port>
  httpsProxy: https://<username>:<password>@<proxy.example.com>:<port>
  noProxy: <wildcard-of-domain>,<provisioning-network/CIDR>,<BMC-address-range/CIDR>
```

   username は、プロキシーサーバーのユーザー名に置き換えます。

   password は、プロキシーサーバーへのアクセス時に使用するパスワードに置き換えます。

   proxy.example.com は、プロキシーサーバーのパスに置き換えます。

   port は、プロキシーサーバーとの通信ポートに置き換えます。

   wildcard-of-domain は、プロキシーをバイパスするドメインのエントリーに置き換えます。

   provisioning-network/CIDR は、プロビジョニングネットワークの IP アドレスと割り当てられた IP アドレスの数 (CIDR 表記) に置き換えます。

   BMC-address-range/CIDR は、BMC アドレスおよびアドレス数 (CIDR 表記) に置き換えます。

   以前の値を追加すると、設定はクラスターに適用されます。

2. クラスターの作成手順を実行してクラスターをプロビジョニングします。**クラスターの作成** を参照してプロバイダーを選択します。

注: install-config YAML は、クラスターをデプロイする場合にのみ使用できます。クラスターをデプロイした後、**install-config.yaml** に加えた新しい変更は適用されません。導入後に設定を更新するには、ポリシーを使用する必要があります。詳細は、**Pod ポリシー** を参照してください。

1.5.4.11.1. 関連情報

- プロバイダーを選択するには、クラスターの作成 を参照してください。
- クラスターのデプロイ後に設定を変更する方法については、Pod ポリシー を参照してください。

- その他のトピックは クラスターライフサイクルの概要 を参照してください。
- プロキシー環境でのクラスターの作成 に戻ります。

1.5.5. クラスターのインポート

別の Kubernetes クラウドプロバイダーからクラスターをインポートできます。インポート後、ターゲットクラスターは multicluster engine Operator ハブクラスターのマネージドクラスターになります。特に指定されていない限りは通常、ハブクラスターとターゲットのマネージドクラスターにアクセスできる場所で、インポートタスクを実行できます。

ハブクラスターは **他** のハブクラスターの管理はできず、自己管理のみが可能です。ハブクラスターは、自動的にインポートして自己管理できるように設定されています。ハブクラスターは手動でインポートする必要はありません。

ハブクラスターを削除して再度インポートする場合は、**local-cluster:true** ラベルを **ManagedCluster** リソースに追加する必要があります。

クラスターを管理できるようにクラスターをインポートする方法についての詳細は、次のトピックを参照してください。

必要なユーザータイプまたはアクセスレベル: クラスター管理者

- コンソールを使用した既存クラスターのインポート
- CLI を使用したマネージドクラスターのインポート
- エージェント登録を使用したマネージドクラスターのインポート
- オンプレミスの Red Hat OpenShift Container Platform クラスターのインポート

1.5.5.1. コンソールを使用したマネージド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。コンソールを使用してマネージドクラスターをインポートする方法については、以下を参照してください。

- 前提条件
- 新規プルシークレットの作成
- クラスターのインポート
- Red Hat OpenShift Dedicated 環境での import コマンドの手動実行
- オプション: クラスター API アドレスの設定
- クラスターの削除

1.5.5.1.1. 前提条件

- デプロイされたハブクラスター。ベアメタルクラスターをインポートする場合は、ハブクラスターを Red Hat OpenShift Container Platform バージョン 4.13 以降にインストールする必要があります。
- 管理するクラスター。

- **base64** コマンドラインツール。
- 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、定義された **multiclusterhub.spec.imagePullSecret**。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。

Required user type or access level:クラスター管理者

1.5.5.1.2. 新規プルシークレットの作成

新しいプルシークレットを作成する必要がある場合は、以下の手順を実行します。

1. cloud.redhat.com から Kubernetes プルシークレットをダウンロードします。
2. プルシークレットをハブクラスターの namespace に追加します。
3. 次のコマンドを実行して、**open-cluster-management** namespace に新しいシークレットを作成します。

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

open-cluster-management をハブクラスターの namespace の名前に置き換えます。ハブクラスターのデフォルトの namespace は **open-cluster-management** です。

path-to-pull-secret を、ダウンロードしたプルシークレットへのパスに置き換えます。

シークレットは、インポート時にマネージドクラスターに自動的にコピーされます。

- 以前にインストールされたエージェントが、インポートするクラスターから削除されていることを確認します。エラーを回避するには、**open-cluster-management-agent** および **open-cluster-management-agent-addon** namespace を削除する必要があります。
- Red Hat OpenShift Dedicated 環境にインポートする場合には、以下の注意点を参照してください。
  - ハブクラスターを Red Hat OpenShift Dedicated 環境にデプロイしている必要があります。
  - Red Hat OpenShift Dedicated のデフォルト権限は dedicated-admin ですが、namespace を作成するための権限がすべて含まれているわけではありません。multicluster engine operator を使用してクラスターをインポートおよび管理するには、**cluster-admin** 権限が必要です。

1.5.5.1.3. クラスターのインポート

利用可能なクラウドプロバイダーごとに、コンソールから既存のクラスターをインポートできます。

注記: ハブクラスターは別のハブクラスターを管理できません。ハブクラスターは、自動的にインポートおよび自己管理するように設定されるため、ハブクラスターを手動でインポートして自己管理する必要はありません。

デフォルトでは、namespace がクラスター名と namespace に使用されますが、これは変更できます。

重要: クラスターを作成すると、コントローラーはクラスターとそのリソースの namespace を作成します。その namespace には、そのクラスターインスタンスのリソースのみを含めるようにしてください。クラスターを破棄すると、namespace とその中のすべてのリソースが削除されます。

マネージドクラスターはすべて、マネージドクラスターセットに関連付けられている必要があります。マネージドクラスターを **ManagedClusterSet** に割り当てない場合は、クラスターは **default** マネージドクラスターセットに自動的に追加されます。

クラスターを別のクラスターセットに追加する場合は、クラスターセットへの **clusterset-admin** 権限が必要です。クラスターのインポート時に **cluster-admin** 権限がない場合は、**clusterset-admin** 権限を持つクラスターセットを選択する必要があります。指定されたクラスターセットに適切な権限がない場合、クラスターのインポートは失敗します。選択するクラスターセットが存在しない場合は、クラスター管理者に連絡して、クラスターセットへの **clusterset-admin** 権限を受け取ってください。

OpenShift Container Platform Dedicated クラスターをインポートし、**vendor=OpenShiftDedicated** のラベルを追加してベンダーを指定しない場合、**vendor=auto-detect** のラベルを追加すると、**managed-by=platform** ラベルがクラスターに自動的に追加されます。この追加されたラベルを使用して、クラスターを OpenShift Container Platform Dedicated クラスターとして識別し、OpenShift Container Platform Dedicated クラスターをグループとして取得できます。

以下の表は、クラスターをインポートする方法を指定する **インポートモード** で使用できるオプションを示しています。

import コマンドの手動実行	Red Hat Ansible Automation Platform テンプレートなど、コンソールで情報を完了および送信した後に、提供されたコマンドをターゲットクラスターで実行してクラスターをインポートします。OpenShift Container Platform D dedicated 環境でクラスターをインポートし、import コマンドを手動で実行する場合は、コンソールに情報を入力する前に OpenShift Container Platform D dedicated 環境でのimport コマンドの手動実行 の手順を完了してください。
既存クラスターのサーバー URL および API トークンを入力します。	インポートするクラスターのサーバー URL および API トークンを指定します。クラスターのアップグレード時に実行する Red Hat Ansible Automation Platform テンプレートを指定できます。
kubeconfig ファイルを指定します。	インポートするクラスターの **kubeconfig** ファイルの内容をコピーして貼り付けます。クラスターのアップグレード時に実行する Red Hat Ansible Automation Platform テンプレートを指定できます。

注記: Ansible Automation Platform ジョブを作成して実行するには、OperatorHub から Red Hat Ansible Automation Platform Resource Operator をインストールし、実行する必要があります。

クラスター API アドレスを設定するには、任意: Configuring the cluster API address を参照してください。

マネージドクラスター klusterlet を特定のノードで実行するように設定するには、オプション: 特定のノードで実行するように klusterlet を設定する を参照してください。

1.5.5.1.3.1. OpenShift Container Platform Dedicated 環境での import コマンドの手動実行

注: Klusterlet OLM Operator と次の手順は非推奨になりました。

OpenShift Container Platform D dedicated 環境にクラスターをインポートし、import コマンドを手動で実行する場合は、追加で手順を完了する必要があります。

1. インポートするクラスターの OpenShift Container Platform コンソールにログインします。
2. インポートするクラスター上に **open-cluster-management-agent** および **open-cluster-management** namespace またはプロジェクトを作成します。
3. OpenShift Container Platform カタログで klusterlet Operator を検索します。
4. 作成した **open-cluster-management** namespace またはプロジェクトに klusterlet Operator をインストールします。
   重要: open-cluster-management-agent namespace に Operator をインストールしないでください。
5. 以下の手順を実行して、import コマンドからブートストラップシークレットをデプロイメントします。
   a. **import-command** という名前で作成したファイルに、import コマンドを貼り付けます。
   b. 以下のコマンドを実行して、新しいファイルにコンテンツを挿入します。

      ```
      cat import-command | awk '{split($0,a,"&&"); print a[3]}' | awk '{split($0,a,"|"); print a[1]}' | sed -e "s/^ echo //" | base64 -d
      ```

   c. 出力で **bootstrap-hub-kubeconfig** という名前のシークレットを見つけ、コピーします。
   d. シークレットをマネージドクラスターの **open-cluster-management-agent** namespace に適用します。
   e. インストールされた Operator の例を使用して klusterlet リソースを作成します。**clusterName** 値をインポート時に設定されたクラスター名と同じ名前に変更します。
      注記: managedcluster リソースがハブに正常に登録されると、2 つの Klusterlet Operator がインストールされます。klusterlet Operator の 1 つは **open-cluster-management** namespace に、もう 1 つは **open-cluster-management-agent** namespace にあります。複数の Operator を使用しても、Klusterlet の機能には影響しません。
6. **Cluster** > **Import cluster** を選択してから、コンソールに情報を提供します。

1.5.5.1.3.2. オプション: クラスター API アドレスの設定

oc get managedcluster コマンドの実行時に表に表示される URL を設定して、クラスターの詳細ページにある **Cluster API アドレス** をオプションで設定します。

1. **cluster-admin** 権限がある ID でハブクラスターにログインします。
2. ターゲットに設定されたマネージドクラスターの **kubeconfig** ファイルを設定します。
3. 次のコマンドを実行して、インポートするクラスターのマネージドクラスターエントリーを編集します。**cluster -name** をマネージドクラスターの名前に置き換えます。

   ```
   oc edit managedcluster <cluster-name>
   ```

4. 以下の例のように、YAML ファイルの **ManagedCluster** 仕様に **managedClusterClientConfigs**

4. 以下の例のように、YAML ファイルの **ManagedCluster** 仕様に **manageClusterClientConfigs** セクションを追加します。

```
spec:
  hubAcceptsClient: true
  managedClusterClientConfigs:
  - url: <https://api.new-managed.dev.redhat.com> (1)
```

(1) URL の値を、インポートするマネージドクラスターへの外部アクセスを提供する URL に置き換えます。

1.5.5.1.3.3. オプション: 特定のノードで実行するように klusterlet を設定する

マネージドクラスターの **nodeSelector** と **tolerations** アノテーションを設定することで、マネージドクラスター klusterlet を実行するノードを指定できます。これらの設定を設定するには、次の手順を実行します。

1. コンソールのクラスターページから、更新するマネージドクラスターを選択します。
2. YAML コンテンツを表示するには、YAML スイッチを **On** に設定します。
   注記: YAML エディターは、クラスターをインポートまたは作成するときにのみ使用できます。インポートまたは作成後にマネージドクラスターの YAML 定義を編集するには、OpenShift Container Platform コマンドラインインターフェイスまたは Red Hat Advanced Cluster Management 検索機能を使用する必要があります。
3. **nodeSelector** アノテーションをマネージドクラスターの YAML 定義に追加します。このアノテーションのキーは、**open-cluster-management/nodeSelector** です。このアノテーションの値は、JSON 形式の文字列マップです。
4. **tolerations** エントリーをマネージドクラスターの YAML 定義に追加します。このアノテーションのキーは、**open-cluster-management/tolerations** です。このアノテーションの値は、JSON 形式の toleration リストを表します。結果の YAML は次の例のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    open-cluster-management/nodeSelector: '{"dedicated":"acm"}'
    open-cluster-management/tolerations:
'[{"key":"dedicated","operator":"Equal","value":"acm","effect":"NoSchedule"}]'
```

KlusterletConfig を使用して、マネージドクラスターの **nodeSelector** と **tolerations** を設定することもできます。これらのオプションを設定するには、次の手順を実行します。

注: KlusterletConfig を使用する場合、マネージドクラスターは、マネージドクラスターのアノテーションの設定ではなく、**KlusterletConfig** 設定の構成を使用します。

1. 次のサンプル YAML の内容を適用します。必要に応じて値を置き換えます。

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: <klusterletconfigName>
spec:
  nodePlacement:
```

```
nodeSelector:
  dedicated: acm
tolerations:
  - key: dedicated
    operator: Equal
    value: acm
    effect: NoSchedule
```

2. **Agent.open-cluster-management.io/klusterlet-config: `<klusterletconfigName>`** アノテーションをマネージドクラスターに追加し、**<klusterletconfigName>** を **KlusterletConfig** の名前に置き換えます。

1.5.5.1.4. インポートされたクラスターの削除

以下の手順を実行して、インポートされたクラスターと、マネージドクラスターで作成された **open-cluster-management-agent-addon** を削除します。

Clusters ページで、 Actions > Detach cluster をクリックしてマネージメントからクラスターを削除します。

注記: local-cluster という名前のハブクラスターをデタッチしようとする場合は、デフォルトの **disableHubSelfManagement** 設定が **false** である点に注意してください。この設定が原因で、ハブクラスターがデタッチされると、自身を再インポートして管理し、**MultiClusterHub** コントローラーが調整されます。ハブクラスターがデタッチプロセスを完了して再インポートするのに時間がかかる場合があります。プロセスが完了するのを待たずにハブクラスターを再インポートする場合は、次のコマンドを実行して **multiclusterhub-operator** Pod を再起動し、再インポートを高速化できます。

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

disableHubSelfManagement の値を **true** に指定して、自動的にインポートされないように、ハブクラスターの値を変更できます。詳細は、**disableHubSelfManagement** のトピックを参照してください。

1.5.5.1.4.1. 関連情報

- カスタムイメージプルシークレットの 定義方法の詳細は、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。
- disableHubSelfManagement トピックを参照してください。

1.5.5.2. CLI を使用したマネージド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、Red Hat OpenShift Container Platform CLI を使用して、クラスターをインポートおよび管理する準備が整います。自動インポートシークレットを使用するか、man コマンドを使用して、CLI でマネージドクラスターをインポートする方法については、以下のトピックを参照してください。

- 前提条件
- サポート対象のアーキテクチャー
- クラスターインポートの準備
- 自動インポートシークレットを使用したクラスターのインポート

- クラスターの手動インポート
- klusterlet アドオンのインポート
- CLI を使用したインポート済みクラスターの削除

重要: ハブクラスターは別のハブクラスターを管理できません。ハブクラスターは、**ローカルクラスター** として自動的にインポートおよび管理されるようにセットアップされます。ハブクラスターは、手動でインポートして自己管理する必要はありません。ハブクラスターを削除して再度インポートする場合は、**local-cluster:true** ラベルを追加する必要があります。

1.5.5.2.1. 前提条件

- デプロイされたハブクラスター。ベアメタルクラスターをインポートする場合は、ハブクラスターを OpenShift Container Platform バージョン 4.13 以降にインストールする必要があります。
- 管理する別のクラスター。
- **oc** コマンドを実行する OpenShift Container Platform CLI バージョン 4.13 以降。OpenShift Container Platform CLI のインストールと設定については、OpenShift CLI の使用方法 を参照してください。
- 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、定義された **multiclusterhub.spec.imagePullSecret**。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。

1.5.5.2.2. サポートされているアーキテクチャー

- Linux (x86_64, s390x, ppc64le)
- MacOS

1.5.5.2.3. クラスターインポートの準備

CLI を使用してマネージドクラスターをインポートする前に、以下の手順を実行する必要があります。

1. 次のコマンドを実行して、ハブクラスターにログインします。

```
oc login
```

2. ハブクラスターで次のコマンドを実行して、プロジェクトおよび namespace を作成します。**<cluster_name>** で定義されているクラスター名は、YAML ファイルとコマンドでクラスター namespace としても使用されます。

```
oc new-project <cluster_name>
```

重要: cluster.open-cluster-management.io/managedCluster ラベルは、マネージドクラスターの namespace に対して自動的に追加および削除されます。手動でマネージドクラスター namespace に追加したり、削除したりしないでください。

3. 以下の内容例で **managed-cluster.yaml** という名前のファイルを作成します。

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

cloud および **vendor** の値を **auto-detect** する場合、Red Hat Advanced Cluster Management はインポートしているクラスターからクラウドおよびベンダータイプを自動的に検出します。オプションで、**auto-detect** の値をクラスターのクラウドおよびベンダーの値に置き換えることができます。以下の例を参照してください。

```
cloud: Amazon
vendor: OpenShift
```

4. 以下のコマンドを実行して、YAML ファイルを **ManagedCluster** リソースに適用します。

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

これで、自動インポートシークレットを使用してクラスターをインポートする か、手動でクラスターをインポートする のいずれかに進むことができます。

1.5.5.2.4. 自動インポートシークレットを使用したクラスターのインポート

自動インポートシークレットを使用してマネージドクラスターをインポートするには、クラスターの **kubeconfig** ファイルへの参照、またはクラスターの kube API サーバーとトークンのペアのいずれかの参照を含むシークレットを作成する必要があります。自動インポートシークレットを使用してクラスターをインポートするには、以下の手順を実行します。

1. インポートするマネージドクラスターの **kubeconfig** ファイル、または kube API サーバーおよび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. ${CLUSTER_NAME} namespace に **auto-import-secret.yaml** ファイルを作成します。

   a. 以下のテンプレートのようなコンテンツを使用して、**auto-import-secret.yaml** という名前の YAML ファイルを作成します。

```
apiVersion: v1
kind: Secret
metadata:
  name: auto-import-secret
  namespace: <cluster_name>
stringData:
  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 token/server pair, add the following two values instead of
  # the kubeconfig file:
```

```
  token: <Token to access the cluster>
  server: <cluster_api_url>
type: Opaque
```

b. 以下のコマンドを実行して、<cluster_name> namespace に YAML ファイルを適用します。

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

注記: デフォルトでは、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が完了すると削除されます。自動インポートシークレットを保持する場合は、**managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret** をシークレットに追加します。これを追加するには、以下のコマンドを実行します。

```
oc -n <cluster_name> annotate secrets auto-import-secret managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret=""
```

3. 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。ハブクラスターから以下のコマンドを実行します。

```
oc get managedcluster <cluster_name>
```

4. クラスターで以下のコマンドを実行して、マネージドクラスターにログインします。

```
oc login
```

5. 以下のコマンドを実行して、インポートするクラスターの Pod ステータスを検証できます。

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

klusterlet アドオンのインポート に進むことができます。

1.5.5.2.5. クラスターの手動インポート

重要: import コマンドには、インポートされた各マネージド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。

マネージドクラスターを手動でインポートするには、以下の手順を実行します。

1. 以下のコマンドを実行して、ハブクラスターでインポートコントローラーによって生成された **klusterlet-crd.yaml** ファイルを取得します。

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

2. 以下のコマンドを実行して、ハブクラスターにインポートコントローラーによって生成された **import.yaml** ファイルを取得します。

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

インポートするクラスターで次の手順を実行します。

3. 次のコマンドを入力して、インポートするマネージドクラスターにログインします。

```
oc login
```

4. 以下のコマンドを実行して、手順 1 で生成した **klusterlet-crd.yaml** を適用します。

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

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

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

6. ハブクラスターから次のコマンドを実行して、インポートするマネージドクラスターの **JOINED** および **AVAILABLE** ステータスを検証できます。

```
oc get managedcluster <cluster_name>
```

klusterlet アドオンのインポート に進むことができます。

1.5.5.2.6. klusterlet アドオンのインポート

KlusterletAddonConfig klusterlet アドオン設定を実装して、マネージドクラスターで他のアドオンを有効にします。次の手順を実行して、設定ファイルを作成して適用します。

1. 以下の例のような YAML ファイルを作成します。

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
  iamPolicyController:
    enabled: true
  policyController:
    enabled: true
  searchCollector:
    enabled: true
```

2. ファイルは **klusterlet-addon-config.yaml** として保存します。

3. 以下のコマンドを実行して YAML を適用します。

```
oc apply -f klusterlet-addon-config.yaml
```

アドオンは、インポートするマネージドクラスターのステータスが **AVAILABLE** になると、インストールされます。

4. 以下のコマンドを実行して、インポートするクラスターのアドオンの Pod ステータスを検証できます。

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

1.5.5.2.7. コマンドラインインターフェイスを使用したインポート済みクラスターの削除

コマンドラインインターフェイスを使用してマネージドクラスターを削除するには、以下のコマンドを実行します。

```
oc delete managedcluster <cluster_name>
```

<cluster_name> をクラスターの名前に置き換えます。

1.5.5.3. エージェント登録を使用したマネージドクラスターのインポート

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すると、クラスターをインポートし、エージェント登録エンドポイントを使用して管理できるようになります。エージェント登録エンドポイントを使用してマネージドクラスターをインポートする方法については、次のトピックをそのまま参照してください。

- 前提条件
- サポート対象のアーキテクチャー
- クラスターのインポート

1.5.5.3.1. 前提条件

- デプロイされたハブクラスター。ベアメタルクラスターをインポートする場合は、ハブクラスターを OpenShift Container Platform バージョン 4.13 以降にインストールする必要があります。
- 管理するクラスター。
- **base64** コマンドラインツール。
- 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、定義された **multiclusterhub.spec.imagePullSecret**。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。
  新規シークレットを作成する必要がある場合は、新規プルシークレットの作成 を参照してください。

1.5.5.3.2. サポート対象のアーキテクチャー

- Linux (x86_64, s390x, ppc64le)
- MacOS

1.5.5.3.3. クラスターのインポート

エージェント登録エンドポイントを使用してマネージドクラスターをインポートするには、次の手順を実行します。

1. ハブクラスターで次のコマンドを実行して、エージェント登録サーバーの URL を取得します。

```
export agent_registration_host=$(oc get route -n multicluster-engine agent-registration -o=jsonpath="{.spec.host}")
```

注: ハブクラスターがクラスター全体のプロキシーを使用している場合は、マネージドクラスターがアクセスできる URL を使用していることを確認してください。

2. 次のコマンドを実行して、cacert を取得します。

```
oc get configmap -n kube-system kube-root-ca.crt -o=jsonpath="{.data['ca\.crt']}" > ca.crt_
```

3. 次の YAML コンテンツを適用して、エージェント登録サーバーが承認するトークンを取得します。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: managed-cluster-import-agent-registration-sa
  namespace: multicluster-engine
---
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: managed-cluster-import-agent-registration-sa-token
  namespace: multicluster-engine
  annotations:
    kubernetes.io/service-account.name: "managed-cluster-import-agent-registration-sa"
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: managedcluster-import-controller-agent-registration-client
rules:
- nonResourceURLs: ["/agent-registration/*"]
  verbs: ["get"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: managed-cluster-import-agent-registration
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: managedcluster-import-controller-agent-registration-client
subjects:
  - kind: ServiceAccount
    name: managed-cluster-import-agent-registration-sa
    namespace: multicluster-engine
```

4. 以下のコマンドを実行してトークンをエクスポートします。

```
export token=$(oc get secret -n multicluster-engine managed-cluster-import-agent-registration-sa-token -o=jsonpath='{.data.token}' | base64 -d)
```

5. 次のコマンドを実行して、自動承認を有効にし、コンテンツを **cluster-manager** にパッチを適用します。

```
oc patch clustermanager cluster-manager --type=merge -p '{"spec":
{"registrationConfiguration":{"featureGates":[
{"feature": "ManagedClusterAutoApproval", "mode": "Enable"}], "autoApproveUsers":
["system:serviceaccount:multicluster-engine:agent-registration-bootstrap"]}}}'
```

注: 自動承認を無効にして、マネージドクラスターからの証明書署名リクエストを手動で承認することもできます。

6. 次のコマンドを実行して、マネージドクラスターに切り替え、cacert を取得します。

```
curl --cacert ca.crt -H "Authorization: Bearer $token" https://$agent_registration_host/agent-registration/crds/v1 | oc apply -f -
```

7. 次のコマンドを実行して、マネージドクラスターをハブクラスターにインポートします。**<clusterName>** は、クラスターの名前に置き換えます。

オプション: <klusterletconfigName> は KlusterletConfig の名前に置き換えます。

```
curl --cacert ca.crt -H "Authorization: Bearer $token" https://$agent_registration_host/agent-registration/manifests/<clusterName>?klusterletconfig=<klusterletconfigName> | oc apply -f -
```

1.5.5.4. オンプレミスの Red Hat OpenShift Container Platform クラスターの手動インポート

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。既存の OpenShift Container Platform クラスターをインポートして、ノードを追加できます。multicluster engine operator をインストールするとハブクラスターが自動的にインポートされるため、手順を完了しなくてもハブクラスターにノードを追加できます。詳細は、次のトピックを引き続きお読みください。

- 前提条件
- クラスターのインポート

1.5.5.4.1. 前提条件

- Central Infrastructure Management サービスの有効化

1.5.5.4.2. クラスターのインポート

静的ネットワークまたはベアメタルホストなしで OpenShift Container Platform クラスターを手動でインポートし、ノードを追加する準備をするには、以下の手順を実行します。

1. 次の YAML コンテンツを適用して、インポートする OpenShift Container Platform クラスターの namespace を作成します。

```
apiVersion: v1
kind: Namespace
metadata:
  name: managed-cluster
```

2. 以下の YAML コンテンツを適用して、インポートしている OpenShift Container Platform クラスターに一致する ClusterImageSet が存在することを確認します。

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  name: openshift-v4.11.18
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-
release@sha256:22e149142517dfccb47be828f012659b1ccf71d26620e6f62468c264a7ce7863
```

3. 次の YAML コンテンツを適用して、イメージにアクセスするためのプルシークレットを追加します。

```
apiVersion: v1
kind: Secret
type: kubernetes.io/dockerconfigjson
metadata:
  name: pull-secret
  namespace: managed-cluster
stringData:
  .dockerconfigjson: <pull-secret-json> 1
```

1 <pull-secret-json> をプルシークレット JSON に置き換えます。

4. **kubeconfig** を OpenShift Container Platform クラスターからハブクラスターにコピーします。

   a. 次のコマンドを実行して、OpenShift Container Platform クラスターから **kubeconfig** を取得します。**kubeconfig** がインポートされるクラスターとして設定されていることを確認します。

   ```
   oc get secret -n openshift-kube-apiserver node-kubeconfigs -ojson | jq '.data["lb-
   ext.kubeconfig"]' --raw-output | base64 -d > /tmp/kubeconfig.some-other-cluster
   ```

   注記: クラスター API にカスタムドメイン経由でアクセスする場合は、まずこの **kubeconfig** を編集して、**certificate-authority-data** フィールドにカスタム証明書を追加し、**server** フィールドをカスタムドメインに合わせて変更する必要があります。

   b. 次のコマンドを実行して、**kubeconfig** をハブクラスターにコピーします。**kubeconfig** がハブクラスターとして設定されていることを確認します。

   ```
   oc -n managed-cluster create secret generic some-other-cluster-admin-kubeconfig --
   from-file=kubeconfig=/tmp/kubeconfig.some-other-cluster
   ```

5. 次の YAML コンテンツを適用して、**AgentClusterInstall** カスタムリソースを作成します。必要に応じて値を置き換えます。

```
apiVersion: extensions.hive.openshift.io/v1beta1
kind: AgentClusterInstall
metadata:
  name: <your-cluster-name> 1
  namespace: <managed-cluster>
spec:
  networking:
```

```
  userManagedNetworking: true
clusterDeploymentRef:
  name: <your-cluster>
imageSetRef:
  name: openshift-v4.11.18
provisionRequirements:
  controlPlaneAgents: 2
sshPublicKey: <""> 3
```

1. クラスターの名前を選択します。
2. シングルノード OpenShift クラスターを使用している場合は、**1** を使用します。マルチノードクラスターを使用している場合は、**3** を使用します。
3. トラブルシューティングのためにノードにログインできるようにします。オプションの **sshPublicKey** フィールドを追加します。

6. 次の YAML コンテンツを適用して、**ClusterDeployment** を作成します。必要に応じて値を置き換えます。

```
apiVersion: hive.openshift.io/v1
kind: ClusterDeployment
metadata:
  name: <your-cluster-name> 1
  namespace: managed-cluster
spec:
  baseDomain: <redhat.com> 2
  installed: <true> 3
  clusterMetadata:
     adminKubeconfigSecretRef:
       name: <your-cluster-name-admin-kubeconfig> 4
     clusterID: <""> 5
     infraID: <""> 6
  clusterInstallRef:
    group: extensions.hive.openshift.io
    kind: AgentClusterInstall
    name: your-cluster-name-install
    version: v1beta1
  clusterName: your-cluster-name
  platform:
    agentBareMetal:
  pullSecretRef:
    name: pull-secret
```

1. クラスターの名前を選択します。
2. **BaseDomain** が OpenShift Container Platform クラスターに使用しているドメインと一致していることを確認してください。
3. OpenShift Container Platform クラスターを実稼働環境のクラスターとして自動的にインポートするには、**true** に設定します。
4. ステップ 4 で作成した **kubeconfig** を参照します。

5 6 実稼働環境では、**clusterID** と **infraID** は空のままにしておきます。

7. 次の YAML コンテンツを適用することで、**InfraEnv** カスタムリソースを追加して、クラスターに追加する新しいホストを検出します。必要に応じて値を置き換えます。
注記: 静的 IP アドレスを使用していない場合、次の例では追加の設定が必要になる場合があります。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: your-infraenv
  namespace: managed-cluster
spec:
  clusterRef:
    name: your-cluster-name
    namespace: managed-cluster
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: ""
```

表1.6 InfraEnv のフィールドの表

フィールド	任意または必須	説明
clusterRef	任意	遅延バインディングを使用している場合、**clusterRef** フィールドはオプションです。遅延バインディングを使用していない場合は、**clusterRef** を追加する必要があります。
sshAuthorizedKey	任意	トラブルシューティングのためにノードにログインできるように、オプションの **sshAuthorizedKey** フィールド

1. インポートに成功すると、ISO ファイルをダウンロードする URL が表示されます。次のコマンドを実行して ISO ファイルをダウンロードします。<url> は表示される URL に置き換えます。
注記: ベアメタルホストを使用すると、ホストの検出を自動化できます。

```
oc get infraenv -n managed-cluster some-other-infraenv -ojson | jq ".status.<url>" --raw-output | xargs curl -k -o /storage0/isos/some-other.iso
```

2. **オプション:** OpenShift Container Platform クラスターで、ポリシーなどの Red Hat Advanced Cluster Management 機能を使用する場合は、**ManagedCluster** リソースを作成します。**ManagedCluster** リソースの名前は、**ClusterDeployment** リソースの名前と一致させてください。**ManagedCluster** リソースがない場合、コンソールではクラスターのステータスが **detached** になります。

1.5.5.5. インポート用のマネージドクラスターでのイメージレジストリーの指定

インポートしているマネージドクラスターのイメージレジストリーを上書きする必要がある場合があります。これには、**ManagedClusterImageRegistry** カスタムリソース定義を作成します。

ManagedClusterImageRegistry カスタムリソース定義は、namespace スコープのリソースです。

ManagedClusterImageRegistry カスタムリソース定義は、Placement が選択するマネージドクラスターのセットを指定しますが、カスタムイメージレジストリーとは異なるイメージが必要になります。マネージドクラスターが新規イメージで更新されると、識別用に各マネージドクラスターに、**open-cluster-management.io/image-registry=<namespace>.<managedClusterImageRegistryName>** のラベルが追加されます。

以下の例は、**ManagedClusterImageRegistry** カスタムリソース定義を示しています。

```
apiVersion: imageregistry.open-cluster-management.io/v1alpha1
kind: ManagedClusterImageRegistry
metadata:
  name: <imageRegistryName>
  namespace: <namespace>
spec:
  placementRef:
    group: cluster.open-cluster-management.io
    resource: placements
    name: <placementName> 1
  pullSecret:
    name: <pullSecretName> 2
  registries: 3
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
```

1 マネージドクラスターのセットを選択するのと同じ namespace 内の配置の名前に置き換えます。

2 カスタムイメージレジストリーからイメージをプルするために使用されるプルシークレットの名前に置き換えます。

3 **ソース** および **ミラー** レジストリーのそれぞれの値をリスト表示します。**mirrored-image-registry-address** および **image-registry-address** は、レジストリーの各 **ミラー** および **ソース** 値に置き換えます。

- 例 1: **registry.redhat.io/rhacm2** という名前のソースイメージレジストリーを **localhost:5000/rhacm2** に、**registry.redhat.io/multicluster-engine** を **localhost:5000/multicluster-engine** に置き換えるには、以下の例を使用します。

```
registries:
- mirror: localhost:5000/rhacm2/
    source: registry.redhat.io/rhacm2
- mirror: localhost:5000/multicluster-engine
    source: registry.redhat.io/multicluster-engine
```

- 例 2: ソースイメージ **registry.redhat.io/rhacm2/registration-rhel8-operator** を **localhost:5000/rhacm2-registration-rhel8-operator** に置き換えるには、以下の例を使用します。

```
registries:
- mirror: localhost:5000/rhacm2-registration-rhel8-operator
    source: registry.redhat.io/rhacm2/registration-rhel8-operator
```

重要: エージェント登録を使用してマネージドクラスターをインポートする場合は、イメージレジストリーを含む **KlusterletConfig** を作成する必要があります。以下の例を参照してください。必要に応じて値を置き換えます。

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: <klusterletconfigName>
spec:
  pullSecret:
    namespace: <pullSecretNamespace>
    name: <pullSecretName>
  registries:
    - mirror: <mirrored-image-registry-address>
      source: <image-registry-address>
    - mirror: <mirrored-image-registry-address>
      source: <image-registry-address>
```

詳細は、エージェント登録エンドポイントを使用したマネージドクラスターのインポート を参照してください。

1.5.5.5.1. ManagedClusterImageRegistry を持つクラスターのインポート

ManagedClusterImageRegistry カスタムリソース定義でカスタマイズされるクラスターをインポートするには、以下の手順を実行します。

1. クラスターをインポートする必要のある namespace にプルシークレットを作成します。これらの手順では、namespace は **myNamespace** です。

   ```
   $ kubectl create secret docker-registry myPullSecret \
     --docker-server=<your-registry-server> \
     --docker-username=<my-name> \
     --docker-password=<my-password>
   ```

2. 作成した namespace に Placement を作成します。

   ```
   apiVersion: cluster.open-cluster-management.io/v1beta1
   kind: Placement
   metadata:
     name: myPlacement
     namespace: myNamespace
   spec:
     clusterSets:
     - myClusterSet
     tolerations:
     - key: "cluster.open-cluster-management.io/unreachable"
       operator: Exists
   ```

   注記: Placement がクラスターを選択できるようにするには、Toleration を **unreachable** に指定する必要があります。

3. **ManagedClusterSet** リソースを作成し、これを namespace にバインドします。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: myClusterSet

---
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  name: myClusterSet
  namespace: myNamespace
spec:
  clusterSet: myClusterSet
```

4. namespace に **ManagedClusterImageRegistry** カスタムリソース定義を作成します。

```
apiVersion: imageregistry.open-cluster-management.io/v1alpha1
kind: ManagedClusterImageRegistry
metadata:
  name: myImageRegistry
  namespace: myNamespace
spec:
  placementRef:
    group: cluster.open-cluster-management.io
    resource: placements
    name: myPlacement
  pullSecret:
    name: myPullSecret
  registry: myRegistryAddress
```

5. コンソールからマネージドクラスターをインポートし、マネージドクラスターセットに追加します。

6. **open-cluster-management.io/image-registry=myNamespace.myImageRegistry** ラベルをマネージドクラスターに追加した後に、マネージドクラスターで import コマンドをコピーして実行します。

1.5.6. クラスターへのアクセス

作成され、管理されている Red Hat OpenShift Container Platform クラスターにアクセスするには、以下の手順を実行します。

1. コンソールから、**Infrastructure** > **Clusters** に移動し、作成したクラスターまたはアクセスするクラスターの名前を選択します。

2. **Reveal credentials** を選択し、クラスターのユーザー名およびパスワードを表示します。クラスターにログインする際に使用するため、この値を書き留めてください。
   注記: インポートしたクラスターでは、**Reveal credentials** オプションは利用できません。

3. クラスターにリンクする **Console URL** を選択します。

4. 手順 3 で確認したユーザー ID およびパスワードを使用して、クラスターにログインします。

1.5.7. マネージドクラスターのスケーリング

作成したクラスターについては、仮想マシンのサイズやノード数など、マネージドクラスターの仕様をカスタマイズおよびサイズ変更できます。インストーラーでプロビジョニングされたインフラストラクチャーをクラスターデプロイメントに使用している場合は、次のオプションを参照してください。

- MachinePool によるスケーリング

クラスターのデプロイメントに中央インフラストラクチャー管理を使用している場合は、次のオプションを参照してください。

- OpenShift Container Platform クラスターへのワーカーノードの追加
- マネージドクラスターへのコントロールプレーンノードの追加

1.5.7.1. MachinePool によるスケーリング

multicluster engine operator を使用してプロビジョニングするクラスターの場合、**MachinePool** リソースが自動的に作成されます。**MachinePool** を使用して、仮想マシンのサイズやノード数など、マネージドクラスターの仕様をさらにカスタマイズおよびサイズ変更できます。

- **MachinePool** リソースの使用は、ベアメタルクラスターではサポートされていません。
- **MachinePool** リソースは、ハブクラスター上の Kubernetes リソースで、 **MachineSet** リソースをマネージドクラスターでグループ化します。
- **MachinePool** リソースは、ゾーンの設定、インスタンスタイプ、ルートストレージなど、マシンリソースのセットを均一に設定します。
- **MachinePool** では、マネージドクラスターで、必要なノード数を手動で設定したり、ノードの自動スケーリングを設定したりするのに役立ちます。

1.5.7.1.1. 自動スケーリングの設定

自動スケーリングを設定すると、トラフィックが少ない場合にリソースをスケールダウンし、多くのリソースが必要な場合に十分にリソースを確保できるようにスケールアップするなど、必要に応じてクラスターに柔軟性を持たせることができます。

- コンソールを使用して **MachinePool** リソースで自動スケーリングを有効にするには、以下の手順を実行します。
  1. ナビゲーションで、**Infrastructure** > **Clusters** を選択します。
  2. ターゲットクラスターの名前をクリックし、**Machine pools** タブを選択します。
  3. マシンプールページのターゲットマシンプールの **Options** メニューから **Enable autoscale** を選択します。
  4. マシンセットレプリカの最小数および最大数を選択します。マシンセットレプリカは、クラスターのノードに直接マップします。
     Scale をクリックした後、変更がコンソールに反映されるまでに数分かかる場合があります。**Machine pools** タブの通知がある場合は、**View machines** をクリックしてスケーリング操作のステータスを表示できます。
- コマンドラインを使用して **MachinePool** リソースで自動スケーリングを有効にするには、以下の手順を実行します。

1. 次のコマンドを入力して、マシンプールのリストを表示します。このとき、**managed-cluster-namespace** は、ターゲットのマネージドクラスターの namespace に置き換えます。

```
oc get machinepools -n <managed-cluster-namespace>
```

2. 以下のコマンドを入力してマシンプールの YAML ファイルを編集します。

```
oc edit machinepool <MachinePool-resource-name> -n <managed-cluster-namespace>
```

   - **MachinePool-resource-name** は **MachinePool** リソースの名前に置き換えます。
   - **managed-cluster-namespace** はマネージドクラスターの namespace の名前に置き換えます。

3. YAML ファイルから **spec.replicas** フィールドを削除します。
4. **spec.autoscaling.minReplicas** 設定および **spec.autoscaling.maxReplicas** フィールドをリソース YAML に追加します。
5. レプリカの最小数を **minReplicas** 設定に追加します。
6. レプリカの最大数を **maxReplicas** 設定に追加します。
7. ファイルを保存して変更を送信します。

1.5.7.1.2. 自動スケーリングの無効化

コンソールまたはコマンドラインを使用して自動スケーリングを無効にできます。

- コンソールを使用して自動スケーリングを無効にするには、次の手順を実行します。
  1. ナビゲーションで、**Infrastructure** > **Clusters** を選択します。
  2. ターゲットクラスターの名前をクリックし、**Machine pools** タブを選択します。
  3. マシンプールページから、ターゲットマシンプールの **Options** メニューから **autoscale** を無効にします。
  4. 必要なマシンセットのレプリカ数を選択します。マシンセットのレプリカは、クラスター上のノードを直接マップします。
     Scale をクリックした後、表示されるまでに数分かかる場合があります。**Machine pools** タブの通知で **View machines** をクリックすると、スケーリングの状態を表示できます。
- コマンドラインを使用して自動スケーリングを無効にするには、以下の手順を実行します。
  1. 以下のコマンドを実行して、マシンプールのリストを表示します。

```
oc get machinepools -n <managed-cluster-namespace>
```

     managed-cluster-namespace は、ターゲットのマネージドクラスターの namespace に置き換えます。

  2. 以下のコマンドを入力してマシンプールの YAML ファイルを編集します。

```
oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
```

name-of-MachinePool-resource は、**MachinePool** リソースの名前に置き換えます。

namespace-of-managed-cluster は、マネージドクラスターの namespace 名に置き換えます。

3. YAML ファイルから **spec.autoscaling** フィールドを削除します。
4. **spec.replicas** フィールドをリソース YAML に追加します。
5. **replicas** の設定にレプリカ数を追加します。
6. ファイルを保存して変更を送信します。

1.5.7.1.3. 手動スケーリングの有効化

コンソールおよびコマンドラインから手動でスケーリングできます。

1.5.7.1.3.1. コンソールでの手動スケーリングの有効化

コンソールを使用して **MachinePool** リソースをスケーリングするには、次の手順を実行します。

1. 有効になっている場合は、**MachinePool** の自動スケーリングを無効にします。前の手順を参照してください。
2. コンソールから、**Infrastructure** > **Clusters** をクリックします。
3. ターゲットクラスターの名前をクリックし、**Machine pools** タブを選択します。
4. マシンプールページで、対象のマシンプールの **Options** メニューから **Scale machine pool** を選択します。
5. 必要なマシンセットのレプリカ数を選択します。マシンセットのレプリカは、クラスター上のノードを直接マップします。**Scale** をクリックした後、変更がコンソールに反映されるまでに数分かかる場合があります。マシンプールタブの通知から **View machines** をクリックすると、スケーリング操作の状態を表示できます。

1.5.7.1.3.2. コマンドラインでの手動スケーリングの有効化

コマンドラインを使用して **MachinePool** リソースをスケーリングするには、次の手順を実行します。

1. 次のコマンドを入力して、マシンプールのリストを表示します。このとき、**<managed-cluster-namespace>** は、ターゲットのマネージドクラスターの namespace に置き換えます。

```
oc get machinepools -n <managed-cluster-namespace>
```

2. 以下のコマンドを入力してマシンプールの YAML ファイルを編集します。

```
oc edit machinepool <MachinePool-resource-name> -n <managed-cluster-namespace>
```

- **MachinePool-resource-name** は **MachinePool** リソースの名前に置き換えます。

- **managed-cluster-namespace** はマネージドクラスターの namespace の名前に置き換えます。

3. YAML ファイルから **spec.autoscaling** フィールドを削除します。
4. YAML ファイルの **spec.replicas** フィールドを必要な数のレプリカに変更します。
5. ファイルを保存して変更を送信します。

1.5.7.1.4. OpenShift Container Platform クラスターへのワーカーノードの追加

実稼働環境のワーカーノードを OpenShift Container Platform クラスターに追加するには、以下の手順を実行します。

1. ワーカーノードとして使用するマシンを、以前にダウンロードした ISO から起動します。
   注記: ワーカーノードが OpenShift Container Platform ワーカーノードの要件を満たしていることを確認してください。
2. 次のコマンドを実行した後、エージェントが登録されるまで待ちます。

```
watch -n 5 "oc get agent -n managed-cluster"
```

3. エージェントの登録が成功すると、エージェントがリストされます。インストールのためにエージェントを承認します。これには数分かかる場合があります。
   注記: エージェントがリストにない場合は、Ctrl キーと C キーを押して **watch** コマンドを終了し、ワーカーノードにログインしてトラブルシューティングを行ってください。
4. 遅延バインディングを使用している場合は、次のコマンドを実行して、保留中のバインドされていないエージェントを OpenShift Container Platform クラスターに関連付けます。遅延バインディングを使用していない場合は、ステップ 5 に進みます。

```
oc get agent -n managed-cluster -ojson | jq -r '.items[] | select(.spec.approved==false)
|select(.spec.clusterDeploymentName==null) | .metadata.name'| xargs oc -n managed-
cluster patch -p '{"spec":{"clusterDeploymentName":{"name":"some-other-
cluster","namespace":"managed-cluster"}}}' --type merge agent
```

5. 次のコマンドを実行して、保留中のエージェントのインストールを承認します。

```
oc get agent -n managed-cluster -ojson | jq -r '.items[] | select(.spec.approved==false) |
.metadata.name'| xargs oc -n managed-cluster patch -p '{"spec":{"approved":true}}' --type
merge agent
```

ワーカーノードのインストールを待ちます。ワーカーノードのインストールが完了すると、ワーカーノードは証明書署名要求 (CSR) を使用してマネージドクラスターに接続し、参加プロセスを開始します。CSR は自動的に署名されます。

1.5.7.2. マネージドクラスターへのコントロールプレーンノードの追加

問題のあるコントロールプレーンを置き換えるには、コントロールプレーンノードを正常または正常でないマネージドクラスターに追加します。

必要なアクセス権限: 管理者

1.5.7.2.1. 正常なマネージドクラスターへのコントロールプレーンノードの追加

以下の手順を実行して、コントロールプレーンノードを正常なマネージドクラスターに追加します。

1. 新規コントロールプレーンノードの OpenShift Container Platform クラスターへのワーカーノードの追加 の手順を実行します。
2. 次のコマンドを実行して、エージェントを承認する前にエージェントを **master** に設定します。

```
oc patch agent <AGENT-NAME> -p '{"spec":{"role": "master"}}' --type=merge
```

   注意: CSR は自動的に承認されません。

3. OpenShift Container Platform の Assisted Installer ドキュメントの 正常なクラスターへのプライマリーコントロールプレーンノードのインストール の手順に従ってください。

1.5.7.2.2. 正常でないマネージドクラスターへのコントロールプレーンノードの追加

以下の手順を実行して、コントロールプレーンノードを正常でないマネージドクラスターに追加します。

1. 正常でないコントロールプレーンノードのエージェントを削除します。
2. デプロイメントにゼロタッチプロビジョニングフローを使用した場合は、ベアメタルホストを削除します。
3. 新規コントロールプレーンノードの OpenShift Container Platform クラスターへのワーカーノードの追加 の手順を実行します。
4. 次のコマンドを実行して、エージェントを承認する前にエージェントを **master** に設定します。

```
oc patch agent <AGENT-NAME> -p '{"spec":{"role": "master"}}' --type=merge
```

   注意: CSR は自動的に承認されません。

5. OpenShift Container Platform の Assisted Installer ドキュメントの 正常でないクラスターへのプライマリーコントロールプレーンノードのインストール の手順に従ってください。

1.5.8. 作成されたクラスターの休止

multicluster engine Operator を使用して作成されたクラスターを休止状態にし、リソースを節約できます。休止状態のクラスターに必要となるリソースは、実行中のものより少なくなるので、クラスターを休止状態にしたり、休止状態を解除したりすることで、プロバイダーのコストを削減できる可能性があります。この機能は、以下の環境の multicluster engine Operator によって作成されたクラスターにのみ適用されます。

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

1.5.8.1. コンソールを使用したクラスターの休止

コンソールを使用して、multicluster engine operator によって作成されたクラスターを休止状態にするには、以下の手順を実行します。

1. ナビゲーションメニューから **Infrastructure** > **Clusters** を選択します。**Manage clusters** タブが選択されていることを確認します。
2. そのクラスターの **Options** メニューから **Hibernate cluster** を選択します。**注記: Hibernate cluster** オプションが利用できない場合には、クラスターを休止状態にすることはできません。これは、クラスターがインポートされ、multicluster engine operator によって作成されていない場合に発生する可能性があります。

Clusters ページのクラスターのステータスは、プロセスが完了すると **Hibernating** になります。

ヒント: Clusters ページで休止するするクラスターを選択し、**Actions** > **Hibernate cluster** を選択して、複数のクラスターを休止できます。

選択したクラスターが休止状態になりました。

1.5.8.2. CLI を使用したクラスターの休止

CLI を使用して multicluster engine operator によって作成されたクラスターを休止状態にするには、以下の手順を実行します。

1. 以下のコマンドを入力して、休止するクラスターの設定を編集します。

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   name-of-cluster は、休止するクラスター名に置き換えます。

   namespace-of-cluster は、休止するクラスターの namespace に置き換えます。

2. **spec.powerState** の値は **Hibernating** に変更します。
3. 以下のコマンドを実行して、クラスターのステータスを表示します。

   ```
   oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
   ```

   name-of-cluster は、休止するクラスター名に置き換えます。

   namespace-of-cluster は、休止するクラスターの namespace に置き換えます。

   クラスターを休止するプロセスが完了すると、クラスターのタイプの値は **type=Hibernating** になります。

選択したクラスターが休止状態になりました。

1.5.8.3. コンソールを使用して休止中のクラスターの通常操作を再開する手順

コンソールを使用して、休止中のクラスターの通常操作を再開するには、以下の手順を実行します。

1. ナビゲーションメニューから **Infrastructure** > **Clusters** を選択します。**Manage clusters** タブが選択されていることを確認します。
2. 再開するクラスターの **Options** メニューから **Resume cluster** を選択します。

プロセスを完了すると、**Clusters** ページのクラスターのステータスは **Ready** になります。

ヒント: Clusters ページで、再開するクラスターを選択し、**Actions** > **Resume cluster** の順に選択して、複数のクラスターを再開できます。

選択したクラスターで通常の操作が再開されました。

1.5.8.4. CLI を使用して休止中のクラスターの通常操作を再開する手順

CLI を使用して、休止中のクラスターの通常操作を再開するには、以下の手順を実行します。

1. 以下のコマンドを入力してクラスターの設定を編集します。

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   name-of-cluster は、休止するクラスター名に置き換えます。

   namespace-of-cluster は、休止するクラスターの namespace に置き換えます。

2. **spec.powerState** の値を **Running** に変更します。

3. 以下のコマンドを実行して、クラスターのステータスを表示します。

   ```
   oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
   ```

   name-of-cluster は、休止するクラスター名に置き換えます。

   namespace-of-cluster は、休止するクラスターの namespace に置き換えます。

   クラスターの再開プロセスが完了すると、クラスターのタイプの値は **type=Running** になります。

選択したクラスターで通常の操作が再開されました。

1.5.9. クラスターのアップグレード

multicluster engine Operator で管理する Red Hat OpenShift Container Platform クラスターを作成した後、multicluster engine Operator コンソールを使用して、マネージドクラスターが使用するバージョンチャネルで利用できる最新のマイナーバージョンにこれらのクラスターをアップグレードできます。

接続された環境では、コンソールでアップグレードが必要な各クラスターに提供される通知によって、更新が自動的に識別されます。

注記:

メジャーバージョンへのアップグレードには、そのバージョンへのアップグレードの前提条件をすべて満たしていることを確認する必要があります。コンソールでクラスターをアップグレードする前に、マネージドクラスターのバージョンチャネルを更新する必要があります。

マネージドクラスターでバージョンチャネルを更新すると、multicluster engine Operator コンソールに、アップグレードに使用できる最新バージョンが表示されます。

このアップグレードの手法は、ステータスが **Ready** の OpenShift Container Platform のマネージドクラスタークラスターでだけ使用できます。

重要: multicluster engine Operator コンソールを使用して、Red Hat OpenShift Kubernetes Service マネージドクラスターまたは OpenShift Container Platform マネージドクラスターを Red Hat OpenShift Dedicated でアップグレードすることはできません。

オンライン環境でクラスターをアップグレードするには、以下の手順を実行します。

1. ナビゲーションメニューから **Infrastructure** > **Clusters** に移動します。アップグレードが利用可能な場合には、**Distribution version** の列に表示されます。
2. アップグレードする **Ready** 状態のクラスターを選択します。クラスターはコンソールを使用してアップグレードされる OpenShift Container Platform クラスターである必要があります。
3. **Upgrade** を選択します。
4. 各クラスターの新しいバージョンを選択します。
5. **Upgrade** を選択します。

クラスターのアップグレードに失敗すると、Operator は通常アップグレードを数回再試行し、停止し、コンポーネントに問題があるステータスを報告します。場合によっては、アップグレードプロセスは、プロセスの完了を繰り返し試行します。アップグレードに失敗した後にクラスターを以前のバージョンにロールバックすることはサポートされていません。クラスターのアップグレードに失敗した場合は、Red Hat サポートにお問い合わせください。

1.5.9.1. チャネルの選択

コンソールを使用して、OpenShift Container Platform でクラスターをアップグレードするためのチャネルを選択できます。チャネルを選択すると、エラータバージョンとリリースバージョンの両方で利用可能なクラスターアップグレードが自動的に通知されます。

クラスターのチャネルを選択するには、以下の手順を実行します。

1. ナビゲーションメニューから **infrastructure** > **Clusters** をクリックします。
2. 変更するクラスターの名前を選択して、**Cluster details** ページを表示します。クラスターに別のチャネルが利用可能な場合は、編集アイコンが **Channel** フィールドに表示されます。
3. 編集アイコンをクリックして、フィールドの設定を変更します。
4. **New channel** フィールドでチャネルを選択します。

利用可能なチャネル更新のリマインダーは、クラスターの **Cluster details** ページで見つけることができます。

1.5.9.2. オフラインクラスターのアップグレード

multicluster engine Operator と Red Hat OpenShift Update Service を使用し、オフライン環境でクラスターをアップグレードできます。

セキュリティー上の理由で、クラスターがインターネットに直接接続できない場合があります。このような場合は、アップグレードが利用可能なタイミングや、これらのアップグレードの処理方法を把握するのが困難になります。OpenShift Update Service を設定すると便利です。

OpenShift Update Service は、個別の Operator および オペランドで、非接続環境で利用可能なマネージドクラスターを監視して、クラスターのアップグレードで利用できるようにします。OpenShift Update Service の設定後に、以下のアクションを実行できます。

- オフラインのクラスター向けにいつアップグレードが利用できるかを監視します。
- グラフデータファイルを使用してアップグレード用にどの更新がローカルサイトにミラーリングされているかを特定します。

- コンソールを使用して、クラスターのアップグレードが利用可能であることを通知します。

次のトピックでは、オフラインクラスターをアップグレードする手順について説明します。

- 前提条件
- 非接続ミラーレジストリーの準備
- OpenShift Update Service の Operator のデプロイ
- グラフデータの init コンテナーの構築
- ミラーリングされたレジストリーの証明書の設定
- OpenShift Update Service インスタンスのデプロイ
- デフォルトのレジストリーの上書き (オプション)
- オフラインカタログソースのデプロイ
- マネージドクラスターのパラメーターを変更する
- 利用可能なアップグレードの表示
- チャネルの選択
- クラスターのアップグレード

1.5.9.2.1. 前提条件

OpenShift Update Service を使用して非接続クラスターをアップグレードするには、以下の前提条件を満たす必要があります。

- 制限付き OLM が設定された Red Hat OpenShift Container Platform バージョン 4.13 以降で実行されているデプロイ済みハブクラスター。制限付きの OLM の設定方法については、ネットワークが制限された環境での Operator Lifecycle Manager の使用 を参照してください。
  Note: 制限付きの OLM の設定時に、カタログソースイメージをメモします。
- ハブクラスターによって管理される OpenShift Container Platform クラスター
- クラスターイメージをミラーリング可能なローカルレジストリーにアクセスするための認証情報。このリポジトリーの作成方法について、詳細は 非接続インストールミラーリング を参照してください。
  注記: アップグレードするクラスターの現行バージョンのイメージは、ミラーリングされたイメージの 1 つとして常に利用可能でなければなりません。アップグレードに失敗すると、クラスターはアップグレード試行時のクラスターのバージョンに戻ります。

1.5.9.2.2. 非接続ミラーレジストリーの準備

ローカルのミラーリングレジストリーに、アップグレード前の現行のイメージと、アップグレード後のイメージの療法をミラーリングする必要があります。イメージをミラーリングするには以下の手順を実行します。

1. 以下の例のような内容を含むスクリプトファイルを作成します。

```
UPSTREAM_REGISTRY=quay.io
PRODUCT_REPO=openshift-release-dev
```

```
RELEASE_NAME=ocp-release
OCP_RELEASE=4.12.2-x86_64
LOCAL_REGISTRY=$(hostname):5000
LOCAL_SECRET_JSON=/path/to/pull/secret 1

oc adm -a ${LOCAL_SECRET_JSON} release mirror \
--
from=${UPSTREAM_REGISTRY}/${PRODUCT_REPO}/${RELEASE_NAME}:${OCP_RELEASE} \
--to=${LOCAL_REGISTRY}/ocp4 \
--to-release-image=${LOCAL_REGISTRY}/ocp4/release:${OCP_RELEASE}
```

1 **path-to-pull-secret** は、OpenShift Container Platform のプルシークレットへのパスに置き換えます。

2. スクリプトを実行して、イメージのミラーリング、設定の設定、リリースイメージとリリースコンテンツの分離を行います。
ImageContentSourcePolicy の作成時に、このスクリプトの最後の行にある出力を使用できます。

1.5.9.2.3. OpenShift Update Service の Operator のデプロイ

OpenShift Container Platform 環境で OpenShift Update Service の Operator をデプロイするには、以下の手順を実行します。

1. ハブクラスターで、OpenShift Container Platform Operator のハブにアクセスします。
2. **Red Hat OpenShift Update Service Operator** を選択して Operator をデプロイします。必要に応じてデフォルト値を更新します。Operator をデプロイすると、**openshift-cincinnati** という名前の新規プロジェクトが作成されます。
3. Operator のインストールが完了するまで待ちます。
OpenShift Container Platform コマンドラインで **oc get pods** コマンドを入力すると、インストールのステータスを確認できます。Operator の状態が **running** であることを確認します。

1.5.9.2.4. グラフデータの init コンテナーの構築

OpenShift Update Service はグラフデータ情報を使用して、利用可能なアップグレードを判別します。オンライン環境では、OpenShift Update Service は Cincinnati グラフデータの GitHub リポジトリー から直接利用可能なアップグレードがないか、グラフデータ情報をプルします。非接続環境を設定しているため、**init container** を使用してローカルリポジトリーでグラフデータを利用できるようにする必要があります。以下の手順を実行して、グラフデータの **init container** を作成します。

1. 以下のコマンドを入力して、**グラフデータ** Git リポジトリーのクローンを作成します。

```
git clone https://github.com/openshift/cincinnati-graph-data
```

2. グラフデータの **init** の情報が含まれるファイルを作成します。このサンプル Dockerfile は、**cincinnati-operator** GitHub リポジトリーにあります。ファイルの内容は以下の例のようになります。

```
FROM registry.access.redhat.com/ubi8/ubi:8.1 1

RUN curl -L -o cincinnati-graph-data.tar.gz https://github.com/openshift/cincinnati-graph-
```

```
data/archive/master.tar.gz 2

RUN mkdir -p /var/lib/cincinnati/graph-data/ 3

CMD exec /bin/bash -c "tar xvzf cincinnati-graph-data.tar.gz -C /var/lib/
cincinnati/graph-data/ --strip-components=1" 4
```

この例では、以下のように設定されています。

1. **FROM** 値は、OpenShift Update Service がイメージを検索する先の外部レジストリーに置き換えます。
2. 3. **RUN** コマンドはディレクトリーを作成し、アップグレードファイルをパッケージ化します。
4. **CMD** コマンドは、パッケージファイルをローカルリポジトリーにコピーして、ファイルをデプロイメントしてアップグレードします。

3. 以下のコマンドを実行して、**graph data init container** をビルドします。

```
podman build -f <path_to_Dockerfile> -t
<${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container>:latest 1 2
podman push <${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container>
<2>:latest --authfile=</path/to/pull_secret>.json 3
```

1. **path_to_Dockerfile** を、前の手順で作成したファイルへのパスに置き換えます。
2. **${DISCONNECTED_REGISTRY}/cincinnati/cincinnati-graph-data-container** は、ローカルグラフデータ init container へのパスに置き換えます。
3. **/path/to/pull_secret** は、プルシークレットへのパスに置き換えます。

注記: podman がインストールされていない場合は、コマンドの **podman** を **docker** に置き換えることもできます。

1.5.9.2.5. ミラーリングされたレジストリーの証明書の設定

セキュアな外部コンテナーレジストリーを使用してミラーリングされた OpenShift Container Platform リリースイメージを保存する場合は、アップグレードグラフをビルドするために OpenShift Update Service からこのレジストリーへのアクセス権が必要です。OpenShift Update Service Pod と連携するように CA 証明書を設定するには、以下の手順を実行します。

1. **image.config.openshift.io** にある OpenShift Container Platform 外部レジストリー API を検索します。これは、外部レジストリーの CA 証明書の保存先です。
   詳細は、OpenShift Container Platform ドキュメントの イメージレジストリーアクセス用の追加のトラストストアの設定 を参照してください。
2. **openshift-config** namespace に ConfigMap を作成します。
3. キー **updateservice-registry** の下に CA 証明書を追加します。OpenShift Update Service はこの設定を使用して、証明書を特定します。

```
apiVersion: v1
kind: ConfigMap
```

```
metadata:
  name: trusted-ca
data:
  updateservice-registry: |
    -----BEGIN CERTIFICATE-----
    ...
    -----END CERTIFICATE-----
```

4. **image.config.openshift.io** API の **cluster** リソースを編集して、**additionalTrustedCA** フィールドを作成した ConfigMap 名に設定します。

```
oc patch image.config.openshift.io cluster -p '{"spec":{"additionalTrustedCA":
{"name":"trusted-ca"}}}' --type merge
```

trusted-ca は、新しい ConfigMap へのパスに置き換えます。

OpenShift Update Service Operator は、変更がないか、**image.config.openshift.io** API と、**openshift-config** namespace に作成した ConfigMap を監視し、CA 証明書が変更された場合はデプロイメントを再起動します。

1.5.9.2.6. OpenShift Update Service インスタンスのデプロイ

ハブクラスターへの OpenShift Update Service インスタンスのデプロイが完了したら、このインスタンスは、クラスターのアップグレードのイメージをミラーリングして非接続マネージドクラスターに提供する場所に配置されます。インスタンスをデプロイするには、以下の手順を実行します。

1. デフォルトの Operator の namespace (**openshift-cincinnati**) を使用しない場合は、お使いの OpenShift Update Service インスタンスの namespace を作成します。
   a. OpenShift Container Platform ハブクラスターコンソールのナビゲーションメニューで、**Administration** > **Namespaces** を選択します。
   b. **Create Namespace** を選択します。
   c. namespace 名と、namespace のその他の情報を追加します。
   d. **Create** を選択して namespace を作成します。
2. OpenShift Container Platform コンソールの **Installed Operators** セクションで、**Red Hat OpenShift Update Service Operator** を選択します。
3. メニューから **Create Instance** を選択します。
4. OpenShift Update Service インスタンスからコンテンツを貼り付けます。YAML ファイルは以下のマニフェストのようになります。

```
apiVersion: cincinnati.openshift.io/v1beta2
kind: Cincinnati
metadata:
  name: openshift-update-service-instance
  namespace: openshift-cincinnati
spec:
  registry: <registry_host_name>:<port> 1
  replicas: 1
  repository: ${LOCAL_REGISTRY}/ocp4/release
  graphDataImage: '<host_name>:<port>/cincinnati-graph-data-container' 2
```

-

1. **spec.registry** の値は、イメージの非接続環境にあるローカルレジストリーへのパスに置き換えます。

2. **spec.graphDataImage** の値は、グラフデータ init container へのパスに置き換えます。これは、**podman push** コマンドを使用して、グラフデータ init container をプッシュする時に使用した値と同じです。

5. **Create** を選択してインスタンスを作成します。

6. ハブクラスター CLI で **oc get pods** コマンドを入力し、インスタンス作成のステータスを表示します。時間がかかる場合がありますが、コマンド結果でインスタンスと Operator が実行中である旨が表示されたらプロセスは完了です。

1.5.9.2.7. デフォルトのレジストリーの上書き (オプション)

注記: 本セクションの手順は、ミラーレジストリーにリリースをミラーリングした場合にのみ該当します。

OpenShift Container Platform にはイメージレジストリーのデフォルト値があり、この値でアップグレードパッケージの検索先を指定します。オフライン環境では、オーバーライドを作成して、その値をリリースイメージをミラーリングしたローカルイメージレジストリーへのパスに置き換えることができます。

デフォルトのレジストリーを上書きするには、次の手順を実行します。

1. 次の内容のような、**mirror.yaml** という名前の YAML ファイルを作成します。

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  name: <your-local-mirror-name> 1
spec:
  repositoryDigestMirrors:
    - mirrors:
        - <your-registry> 2
      source: registry.redhat.io
```

1. **your-local-mirror-name** をローカルミラーの名前に置き換えます。

2. **your-registry** をローカルミラーリポジトリーへのパスに置き換えます。

   注記: oc adm release mirror コマンドを入力すると、ローカルミラーへのパスが分かります。

2. マネージドクラスターのコマンドラインを使用して、次のコマンドを実行してデフォルトのレジストリーをオーバーライドします。

```
oc apply -f mirror.yaml
```

1.5.9.2.8. オフラインカタログソースのデプロイ

マネージドクラスターで、デフォルトのカタログソースをすべて無効にし、新しいカタログソースを作成します。デフォルトの場所を接続された場所からオフラインローカルレジストリーに変更するには、次の手順を実行します。

1. 次の内容のような、**source.yaml** という名前の YAML ファイルを作成します。

```
apiVersion: config.openshift.io/v1
kind: OperatorHub
metadata:
  name: cluster
spec:
  disableAllDefaultSources: true

---
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-operator-catalog
  namespace: openshift-marketplace
spec:
  sourceType: grpc
  image: '<registry_host_name>:<port>/olm/redhat-operators:v1' 1
  displayName: My Operator Catalog
  publisher: grpc
```

1 **spec.image** の値を、ローカルの制約付きカタログソースイメージへのパスに置き換えます。

2. マネージドクラスターのコマンドラインで、次のコマンドを実行してカタログソースを変更します。

```
oc apply -f source.yaml
```

1.5.9.2.9. マネージドクラスターのパラメーターを変更する

マネージドクラスターの **ClusterVersion** リソース情報を更新して、アップグレードを取得するデフォルトの場所を変更します。

1. マネージドクラスターから、以下のコマンドを入力して **ClusterVersion** アップストリームパラメーターがデフォルトの OpenShift Update Service オペランドであることを確認します。

```
oc get clusterversion -o yaml
```

返される内容は以下のようになります。

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
[..]
  spec:
    channel: stable-4.13
    upstream: https://api.openshift.com/api/upgrades_info/v1/graph
```

2. ハブクラスターから、次のコマンドを入力して、OpenShift Update Service オペランドへのルート URL を特定します。

```
oc get routes
```

後のステップのために戻り値に注意してください。

3. マネージドクラスターのコマンドラインで、次のコマンドを入力して **ClusterVersion** リソースを編集します。

```
oc edit clusterversion version
```

spec.channel の値を新しいバージョンに置き換えます。

spec.upstream の値は、ハブクラスター OpenShift Update Service オペランドへのパスに置き換えます。次の手順を実行して、オペランドへのパスを決定できます。

a. ハブクラスターで以下のコマンドを実行してください。

```
oc get routes -A
```

b. **cincinnati** へのパスを見つけます。オペランドのパスは、**HOST/PORT** フィールドの値です。

4. マネージドクラスターのコマンドラインで、次のコマンドを入力して、**ClusterVersion** のアップストリームパラメーターがローカルハブクラスターの OpenShift Update Service URL で更新されていることを確認します。

```
oc get clusterversion -o yaml
```

結果は次のような内容になります。

```
apiVersion: v1
items:
- apiVersion: config.openshift.io/v1
  kind: ClusterVersion
[..]
  spec:
    channel: stable-4.13
    upstream: https://<hub-cincinnati-uri>/api/upgrades_info/v1/graph
```

1.5.9.2.10. 利用可能なアップグレードの表示

Cluster ページでは、オフラインレジストリーにアップグレードがある場合、クラスターの **Distribution version** に、利用可能なアップグレードがあることが示されます。クラスターを選択し、**Actions** メニューから **Upgrade clusters** を選択すると、利用可能なアップグレードを表示できます。オプションのアップグレードパスが利用可能な場合は、利用可能なアップグレードがリストされます。

注記: 現行バージョンがローカルのイメージリポジトリーにミラーリングされていないと、利用可能なアップグレードバージョンは表示されません。

1.5.9.2.11. チャネルの選択

コンソールを使用して、OpenShift Container Platform バージョン 4.6 以降でクラスターをアップグレードするためのチャネルを選択できます。これらのバージョンはミラーレジストリーで利用可能である必要があります。Selecting a channel の手順を実行して、アップグレードチャネルを指定します。

1.5.9.2.12. クラスターのアップグレード

オフラインレジストリーを設定すると、Multi-Cluster Engine Operator と OpenShift Update Service はオフラインレジストリーを使用して、アップグレードが利用可能かどうかを判断します。利用可能なアップグレードが表示されない場合は、クラスターの現行のリリースイメージと、1 つ後のイメージがローカルリポジトリーにミラーリングされていることを確認します。クラスターの現行バージョンのリリースイメージが利用できないと、アップグレードは利用できません。

Cluster ページでは、オフラインレジストリーにアップグレードがある場合、クラスターの **Distribution version** に、利用可能なアップグレードがあることが示されます。イメージをアップグレードするには、**Upgrade available** をクリックし、アップグレードするバージョンを選択します。

マネージドクラスターは、選択したバージョンに更新されます。

クラスターのアップグレードに失敗すると、Operator は通常アップグレードを数回再試行し、停止し、コンポーネントに問題があるステータスを報告します。場合によっては、アップグレードプロセスは、プロセスの完了を繰り返し試行します。アップグレードに失敗した後にクラスターを以前のバージョンにロールバックすることはサポートされていません。クラスターのアップグレードに失敗した場合は、Red Hat サポートにお問い合わせください。

1.5.10. クラスタープロキシーアドオンの使用

一部の環境では、マネージドクラスターはファイアウォールの背後にあり、ハブクラスターから直接アクセスすることはできません。アクセスを取得するには、プロキシーアドオンを設定してマネージドクラスターの **kube-apiserver** にアクセスし、よりセキュアな接続を提供できます。

重要: ハブクラスター上にクラスター全体のプロキシー設定を配置しないようにしてください。

必要なアクセス権限: 編集

ハブクラスターとマネージドクラスターのクラスタープロキシーアドオンを設定するには、次の手順を実行します。

1. 次の手順を実行してマネージドクラスター **kube-apiserver** にアクセスするように **kubeconfig** ファイルを設定します。

   a. マネージドクラスターに有効なアクセストークンを指定します。
   注記: サービスアカウントの対応するトークンを使用できます。デフォルトの namespace にあるデフォルトのサービスアカウントを使用することもできます。

      i. 次のコマンドを実行して、マネージドクラスターの **kubeconfig** ファイルをエクスポートします。

      ```
      export KUBECONFIG=<managed-cluster-kubeconfig>
      ```

      ii. 次のコマンドを実行して、Pod へのアクセス権があるロールをサービスアカウントに追加します。

      ```
      oc create role -n default test-role --verb=list,get --resource=pods
      oc create rolebinding -n default test-rolebinding --serviceaccount=default:default --role=test-role
      ```

      iii. 次のコマンドを実行して、サービスアカウントトークンのシークレットを見つけます。

      ```
      oc get secret -n default | grep <default-token>
      ```

default-token は、シークレット名に置き換えます。

iv. 次のコマンドを実行して、トークンをコピーします。

```
export MANAGED_CLUSTER_TOKEN=$(kubectl -n default get secret <default-token> -o jsonpath={.data.token} | base64 -d)
```

default-token は、シークレット名に置き換えます。

b. Red Hat Advanced Cluster Management ハブクラスターで **kubeconfig** ファイルを設定します。

i. 次のコマンドを実行して、ハブクラスター上の現在の **kubeconfig** ファイルをエクスポートします。

```
oc config view --minify --raw=true > cluster-proxy.kubeconfig
```

ii. エディターで **server** ファイルを変更します。この例では、**sed** の使用時にコマンドを使用します。OSX を使用している場合は、**alias sed=gsed** を実行します。

```
export TARGET_MANAGED_CLUSTER=<managed-cluster-name>

export NEW_SERVER=https://$(oc get route -n multicluster-engine cluster-proxy-addon-user -o=jsonpath='{.spec.host}')/$TARGET_MANAGED_CLUSTER

sed -i'' -e '/server:/c\    server: "'$NEW_SERVER'"' cluster-proxy.kubeconfig

export CADATA=$(oc get configmap -n openshift-service-ca kube-root-ca.crt -o=go-template='{{index .data "ca.crt"}}' | base64)

sed -i'' -e '/certificate-authority-data:/c\    certificate-authority-data: "'$CADATA'"' cluster-proxy.kubeconfig
```

iii. 次のコマンドを入力して、元のユーザー認証情報を削除します。

```
sed -i'' -e '/client-certificate-data/d' cluster-proxy.kubeconfig
sed -i'' -e '/client-key-data/d' cluster-proxy.kubeconfig
sed -i'' -e '/token/d' cluster-proxy.kubeconfig
```

iv. サービスアカウントのトークンを追加します。

```
sed -i'' -e '$a\    token: "'$MANAGED_CLUSTER_TOKEN'"' cluster-proxy.kubeconfig
```

2. 次のコマンドを実行して、ターゲットマネージドクラスターのターゲット namespace にあるすべての Pod をリスト表示します。

```
oc get pods --kubeconfig=cluster-proxy.kubeconfig -n <default>
```

default namespace は、使用する namespace に置き換えます。

3. マネージドクラスター上の他のサービスにアクセスします。この機能は、マネージドクラスターが Red Hat OpenShift Container Platform クラスターの場合に利用できます。サービスは **service-serving-certificate** を使用してサーバー証明書を生成する必要があります。

- マネージドクラスターから、以下のサービスアカウントトークンを使用します。

```
export PROMETHEUS_TOKEN=$(kubectl get secret -n openshift-monitoring $(kubectl
get serviceaccount -n openshift-monitoring prometheus-k8s -
o=jsonpath='{.secrets[0].name}') -o=jsonpath='{.data.token}' | base64 -d)
```

- ハブクラスターから、以下のコマンドを実行して認証局をファイルに変換します。

```
oc get configmap kube-root-ca.crt -o=jsonpath='{.data.ca\.crt}' > hub-ca.crt
```

4. 以下のコマンドを使用して、マネージドクラスターの Prometheus メトリックを取得します。

```
export SERVICE_NAMESPACE=openshift-monitoring
export SERVICE_NAME=prometheus-k8s
export SERVICE_PORT=9091
export SERVICE_PATH="api/v1/query?query=machine_cpu_sockets"
curl --cacert hub-ca.crt
$NEW_SERVER/api/v1/namespaces/$SERVICE_NAMESPACE/services/$SERVICE_NAME:$
SERVICE_PORT/proxy-service/$SERVICE_PATH -H "Authorization: Bearer
$PROMETHEUS_TOKEN"
```

1.5.10.1. クラスタープロキシーアドオンのプロキシー設定

マネージドクラスターが HTTP および HTTPS プロキシーサーバー経由でハブクラスターと通信できるように、クラスタープロキシーアドオンのプロキシー設定を指定できます。クラスタープロキシーアドオンエージェントがプロキシーサーバー経由でハブクラスターにアクセスする必要がある場合は、プロキシー設定が必要な可能性があります。

クラスタープロキシーアドオンのプロキシー設定を指定するには、次の手順を実行します。

1. ハブクラスターに **AddOnDeploymentConfig** リソースを作成し、**spec.proxyConfig** パラメーターを追加します。以下の例を参照してください。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: <name> 1
  namespace: <namespace> 2
spec:
  agentInstallNamespace: open-cluster-managment-addon-observability
  proxyConfig:
  httpsProxy: "http://<username>:<password>@<ip>:<port>" 3
  noProxy: ".cluster.local,.svc,172.30.0.1" 4
  caBundle: <value> 5
```

1. アドオンのデプロイメント設定名を追加します。
2. マネージドクラスター名を追加します。
3. HTTP プロキシーまたは HTTPS プロキシーのいずれかを指定します。
4. **kube-apiserver** の IP アドレスを追加します。IP アドレスを取得するには、マネージドクラスターでコマンド **oc -ndefaultdescribesvckubernetes | grep IP:** を実行します。
5. **httpsProxy** フィールドで HTTPS プロキシーを指定する場合は、プロキシーサーバー CA バンドルを設定します。

2. 作成した **AddOnDeploymentConfig** リソースを参照して、**ManagedClusterAddOn** リソースを更新します。以下の例を参照してください。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: cluster-proxy
  namespace: <namespace> 1
spec:
installNamespace: open-cluster-managment-addon
configs:
  group: addon.open-cluster-management.io
  resource: AddonDeploymentConfig
  name: <name> 2
  namespace: <namespace> 3
```

1 マネージドクラスター名を追加します。

2 アドオンのデプロイメント設定名を追加します。

3 マネージドクラスター名を追加します。

1. **open-cluster-managment-addon** namespace 内のクラスタープロキシーエージェント Pod に、マネージドクラスターの **HTTPS_PROXY** または **NO_PROXY** 環境変数があるかどうかを確認して、プロキシー設定を確認します。

1.5.11. マネージドクラスターで実行する Ansible Automation Platform タスクの設定

multicluster engine operator は Red Hat Ansible Automation Platform と統合されるため、クラスターの作成またはアップグレードの前後に発生するプリフックとポストフックの Ansible ジョブインスタンスを作成できます。クラスター破棄の prehook および posthook ジョブの設定やクラスターのスケールアクションはサポートされません。

必要なアクセス権限: クラスターの管理者

- 前提条件
- コンソールを使用して、クラスターで実行するように、自動化テンプレートを設定する
- 自動化テンプレートの作成
- Ansible ジョブのステータスの表示

1.5.11.1. 前提条件

クラスターで自動化テンプレートを実行するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform 4.13 以降
- Ansible Automation Platform Resource Operator をインストールして、Ansible ジョブを Git サブスクリプションのライフサイクルに接続している。自動化テンプレートを使用して Ansible Automation Platform ジョブを起動する場合に最良の結果を得るには、Ansible Automation

Platform ジョブテンプレートを実行時にべき等にする必要があります。Ansible Automation Platform Resource Operator は、Red Hat OpenShift Container Platform **OperatorHub** ページから検索できます。

1.5.11.2. コンソールを使用して、クラスターで実行するように、自動化テンプレートを設定する

クラスターの作成時、クラスターのインポート時、またはクラスターの作成後、クラスターに使用する自動化テンプレートを指定できます。

クラスターの作成時またはインポート時にテンプレートを指定するには、**Automation** の手順でクラスターに適用する Ansible テンプレートを選択します。自動化テンプレートがない場合は、**Add automation template** をクリックして作成します。

クラスターの作成後にテンプレートを指定するには、既存のクラスターのアクションメニューで **Update automation template** をクリックします。**Update automation template** オプションを使用して、既存の自動化テンプレートを更新することもできます。

1.5.11.3. 自動化テンプレートの作成

クラスターのインストールまたはアップグレードで Ansible ジョブを開始するには、自動化テンプレートを作成して、いつジョブを実行するかを指定する必要があります。これらは、クラスターのインストールまたはアップグレード前後に実行するように設定できます。

テンプレートの作成時に Ansible テンプレートの実行に関する詳細を指定するには、コンソールで以下の手順を実行します。

1. ナビゲーションから **Infrastructure** > **Automation** を選択します。
2. 状況に適したパスを選択します。
   - 新規テンプレートを作成する場合には、**Create Ansible template** をクリックして手順 3 に進みます。
   - 既存のテンプレートを変更する場合は、変更するテンプレートの **Options** メニューの **Edit template** をクリックして、手順 5 に進みます。
3. 一意のテンプレート名を入力します。名前には小文字の英数字またはハイフン (-) を指定してください。
4. 新規テンプレートの認証情報を選択します。
5. 認証情報を選択した後、すべてのジョブに使用する Ansible インベントリーを選択できます。Ansible 認証情報を Ansible テンプレートにリンクするには、以下の手順を実行します。
   a. ナビゲーションから **Automation** を選択します。認証情報にリンクされていないテンプレートの一覧内のテンプレートには、テンプレートを既存の認証情報にリンクするために使用できるリンク **認証情報へのリンク** アイコンが含まれています。テンプレートと同じ namespace の認証情報のみが表示されます。
   b. 選択できる認証情報がない場合や、既存の認証情報を使用しない場合は、リンクするテンプレートの **Options** メニューから **Edit template** を選択します。
   c. 認証情報を作成する場合は、**Add credential** をクリックして、Ansible Automation Platform の認証情報の作成 の手順を行います。
   d. テンプレートと同じ namespace に認証情報を作成したら、テンプレートの編集時に **Ansible Automation Platform credential** フィールドで認証情報を選択します。

6. クラスターをインストールする前に、Ansible ジョブを開始する場合は、**Pre-install** **Automation templates** セクションの **Add an Automation template** を選択します。

7. 表示されるモーダルで **Job template** または **Workflow job template** を選択します。**job_tags**、**Skip_tags**、およびワークフロータイプを追加することもできます。

   - **Extra variables** フィールドを使用して、**キー=値** ペアの形式でデータを **AnsibleJob** リソースに渡します。

   - 特殊キーの **cluster_deployment** と **install_config** は、追加の変数として自動的に渡されます。こちらには、クラスターに関する一般情報とクラスターのインストール設定に関する詳細が含まれています。

8. クラスターのインストールまたはアップグレードに追加するプリフックおよびポストフックの Ansible ジョブの名前を選択します。

9. 必要に応じて、Ansible ジョブをドラッグして、順番を変更します。

10. **インストール後の自動化テンプレート** セクション、**アップグレード前の自動化テンプレート** セクション、および **アップグレード後の自動化テンプレート** セクションでクラスターのインストール後に開始するすべての自動化テンプレートに手順 5 ～ 7 を繰り返します。クラスターをアップグレードする場合、**Extra variables** フィールドを使用して、**key=value** ペアの形式で **AnsibleJob** リソースにデータを渡すことができます。**cluster_deployment** 特殊キーおよび **install_config** 特殊キーに加えて、**cluster_info** 特殊キーも、**ManagedClusterInfo** リソースからのデータを含む追加変数として自動的に渡されます。

Ansible テンプレートは、指定のアクションが起こるタイミングで、このテンプレートを指定するクラスターで実行するように設定されます。

1.5.11.4. Ansible ジョブのステータスの表示

実行中の Ansible ジョブのステータスを表示して、起動し、正常に実行されていることを確認できます。実行中の Ansible ジョブの現在のステータスを表示するには、以下の手順を実行します。

1. メニューで、**Infrastructure** > **Clusters** を選択して、**Clusters** ページにアクセスします。

2. クラスターの名前を選択して、その詳細を表示します。

3. クラスター情報で Ansible ジョブの最後の実行ステータスを表示します。エントリーには、以下のステータスの 1 つが表示されます。

   - インストール prehook または posthook ジョブが失敗すると、クラスターのステータスは **Failed** と表示されます。

   - アップグレード prehook または posthook ジョブが失敗すると、アップグレードに失敗した **Distribution** フィールドに警告が表示されます。

1.5.11.5. 失敗した Ansible ジョブを再度実行する

クラスターの prehook または posthook が失敗した場合は、**Clusters** ページからアップグレードを再試行できます。

時間を節約するために、クラスター自動化テンプレートの一部である失敗した Ansible ポストフックのみを実行できるようになりました。アップグレード全体を再試行せずに、ポストフックのみを再度実行するには、次の手順を実行します。

1. 次のコンテンツを **ClusterCurator** リソースのルートに追加して、インストールポストフックを再度実行します。

```
operation:
  retryPosthook: installPosthook
```

2. 次のコンテンツを **ClusterCurator** リソースのルートに追加して、アップグレードポストフックを再度実行します。

```
operation:
  retryPosthook: upgradePosthook
```

コンテンツを追加すると、Ansible ポストフックを実行するための新しいジョブが作成されます。

1.5.11.6. すべてのジョブに使用する Ansible インベントリーの指定

ClusterCurator リソースを使用して、すべてのジョブに使用する Ansible インベントリーを指定できます。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: test-inno
  namespace: test-inno
spec:
  desiredCuration: upgrade
  destroy: {}
  install: {}
  scale: {}
  upgrade:
    channel: stable-4.13
    desiredUpdate: 4.13.1
    monitorTimeout: 150
    posthook:
    - extra_vars: {}
      clusterName: test-inno
      type: post_check
      name: ACM Upgrade Checks
    prehook:
    - extra_vars: {}
      clusterName: test-inno
      type: pre_check
      name: ACM Upgrade Checks
    towerAuthSecret: awx
```

インベントリーが作成されたことを確認するには、**ClusterCurator** リソースの **status** フィールドで、すべてのジョブが正常に完了したことを示すメッセージを確認します。

1.5.12. Ansible Automation Platform ジョブをホステッドクラスター上で実行されるように設定

Red Hat Ansible Automation Platform は multicluster engine Operator と統合されているため、ホステッドクラスターの作成または更新の前後に発生するプリフックおよびポストフックの Ansible Automation Platform ジョブインスタンスを作成できます。

必要なアクセス権限: クラスターの管理者

- 前提条件
- Ansible Automation Platform ジョブを実行してホステッドクラスターをインストールする
- Ansible Automation Platform ジョブを実行してホステッドクラスターを更新する
- Ansible Automation Platform ジョブを実行してホステッドクラスターを削除する

1.5.12.1. 前提条件

クラスターで自動化テンプレートを実行するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform 4.14 以降
- Ansible Automation Platform Resource Operator をインストールして、Ansible Automation Platform ジョブを Git サブスクリプションのライフサイクルに接続する。Automation テンプレートを使用して Ansible Automation Platform ジョブを開始する場合は、実行時に Ansible Automation Platform ジョブテンプレートが冪等である。Ansible Automation Platform Resource Operator は、Red Hat OpenShift Container Platform **OperatorHub** ページから検索できます。

1.5.12.2. Ansible Automation Platform ジョブを実行してホステッドクラスターをインストールする

ホステッドクラスターをインストールする Ansible Automation Platform ジョブを開始するには、次の手順を実行します。

1. **pausedUntil: true** フィールドを含む **HostedCluster** および **NodePool** リソースを作成します。**hcp create cluster** コマンドラインインターフェイスコマンドを使用する場合は、**--pausedUntil: true** フラグを指定できます。
   以下の例を参照してください。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: my-cluster
  namespace: clusters
spec:
  pausedUntil: 'true'
...
```

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  name: my-cluster-us-east-2
  namespace: clusters
spec:
  pausedUntil: 'true'
...
```

2. **HostedCluster** リソースと同じ名前と **HostedCluster** リソースと同じ namespace で、**ClusterCurator** リソースを作成します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
  desiredCuration: install
  install:
    jobMonitorTimeout: 5
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
          variable2: 2
      - name: Demo Job Template
    posthook:
      - name: Demo Job Template
    towerAuthSecret: toweraccess
```

3. Ansible Automation Platform Tower で認証が必要な場合は、シークレットリソースを作成します。以下の例を参照してください。

```
apiVersion: v1
kind: Secret
metadata:
  name: toweraccess
  namespace: clusters
stringData:
  host: https://my-tower-domain.io
  token: ANSIBLE_TOKEN_FOR_admin
```

1.5.12.3. Ansible Automation Platform ジョブを実行してホステッドクラスターを更新する

ホステッドクラスターを更新する Ansible Automation Platform ジョブを実行するには、更新するホステッドクラスターの **ClusterCurator** リソースを編集します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
  desiredCuration: upgrade
  upgrade:
    desiredUpdate: 4.14.1 (1)
    monitorTimeout: 120
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
```

```
        variable2: 2
    - name: Demo Job Template
  posthook:
    - name: Demo Job Template
  towerAuthSecret: toweraccess
```

1 サポート対象バージョンの詳細は、**Hosted control plane** を参照してください。

注: この方法でホステッドクラスターを更新すると、Hosted control plane とノードプールの両方が同じバージョンに更新されます。Hosted control plane とノードプールを別のバージョンに更新することはサポートされていません。

1.5.12.4. Ansible Automation Platform ジョブを実行してホステッドクラスターを削除する

ホステッドクラスターを削除する Ansible Automation Platform ジョブを実行するには、削除するホステッドクラスターの **ClusterCurator** リソースを編集します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ClusterCurator
metadata:
  name: my-cluster
  namespace: clusters
  labels:
    open-cluster-management: curator
spec:
  desiredCuration: destroy
  destroy:
    jobMonitorTimeout: 5
    prehook:
      - name: Demo Job Template
        extra_vars:
          variable1: something-interesting
          variable2: 2
      - name: Demo Job Template
    posthook:
      - name: Demo Job Template
    towerAuthSecret: toweraccess
```

注: AWS でホストされているクラスターの削除はサポートされていません。

1.5.12.5. 関連情報

- Hosted control plane コマンドラインインターフェイス (**hcp**) の詳細は、Hosted control plane コマンドラインインターフェイスのインストール を参照してください。
- サポートされているバージョンなど、ホステッドクラスターの詳細は、Hosted control plane を参照してください。

1.5.13. ClusterClaims

ClusterClaim は、マネージドクラスター上のカスタムリソース定義 (CRD) です。ClusterClaim は、マネージドクラスターが要求する情報の一部を表します。ClusterClaim を使用して、ターゲットクラスターでのリソースの placement を解除できます。

以下の例は、YAML ファイルで特定された ClusterClaim を示しています。

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: id.openshift.io
spec:
  value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
```

次の表は、multicluster engine Operator が管理するクラスターにある可能性がある定義済みの ClusterClaim を示しています。

要求名	予約	変更可能	説明
id.k8s.io	true	false	アップストリームの提案で定義された ClusterID
kubeversion.open-cluster-management.io	true	true	Kubernetes バージョン
platform.open-cluster-management.io	true	false	AWS、GCE、Equinix Metal など、マネージドクラスターが稼働しているプラットフォーム
product.open-cluster-management.io	true	false	OpenShift、Anthos、EKS、および GKE などの製品名
id.openshift.io	false	false	OpenShift Container Platform クラスターでのみ利用できる OpenShift Container Platform 外部 ID
consoleurl.openshift.io	false	true	OpenShift Container Platform クラスターでのみ利用できる管理コンソールの URL
version.openshift.io	false	true	OpenShift Container Platform クラスターでのみ利用できる OpenShift Container Platform バージョン

マネージドクラスターで以前の要求が削除されるか、更新されると、自動的に復元またはロールバックされます。

マネージドクラスターがハブに参加すると、マネージドクラスターで作成された ClusterClaim は、ハブの **ManagedCluster** リソースのステータスと同期されます。ClusterClaims のマネージドクラスターは、以下の例のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    cloud: Amazon
    clusterID: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    installer.name: multiclusterhub
    installer.namespace: open-cluster-management
    name: cluster1
    vendor: OpenShift
  name: cluster1
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
status:
  allocatable:
    cpu: '15'
    memory: 65257Mi
  capacity:
    cpu: '18'
    memory: 72001Mi
  clusterClaims:
    - name: id.k8s.io
      value: cluster1
    - name: kubeversion.open-cluster-management.io
      value: v1.18.3+6c42de8
    - name: platform.open-cluster-management.io
      value: AWS
    - name: product.open-cluster-management.io
      value: OpenShift
    - name: id.openshift.io
      value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
    - name: consoleurl.openshift.io
      value: 'https://console-openshift-console.apps.xxxx.dev04.red-chesterfield.com'
    - name: version.openshift.io
      value: '4.13'
  conditions:
    - lastTransitionTime: '2020-10-26T07:08:49Z'
      message: Accepted by hub cluster admin
      reason: HubClusterAdminAccepted
      status: 'True'
      type: HubAcceptedManagedCluster
    - lastTransitionTime: '2020-10-26T07:09:18Z'
      message: Managed cluster joined
      reason: ManagedClusterJoined
      status: 'True'
      type: ManagedClusterJoined
    - lastTransitionTime: '2020-10-30T07:20:20Z'
      message: Managed cluster is available
      reason: ManagedClusterAvailable
      status: 'True'
```

```
    type: ManagedClusterConditionAvailable
  version:
    kubernetes: v1.18.3+6c42de8
```

1.5.13.1. 既存の ClusterClaim の表示

kubectl コマンドを使用して、マネージドクラスターに適用される ClusterClaim をリスト表示できます。これは、ClusterClaim をエラーメッセージと比較する場合に便利です。

注記: clusterclaims.cluster.open-cluster-management.io のリソースに **list** の権限があることを確認します。

以下のコマンドを実行して、マネージドクラスターにある既存の ClusterClaim のリストを表示します。

```
kubectl get clusterclaims.cluster.open-cluster-management.io
```

1.5.13.2. カスタム ClusterClaims の作成

ClusterClaim は、マネージドクラスターでカスタム名を使用して作成できるため、簡単に識別できます。カスタム ClusterClaim は、ハブクラスターの **ManagedCluster** リソースのステータスと同期されます。以下のコンテンツでは、カスタマイズされた **ClusterClaim** の定義例を示しています。

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: <custom_claim_name>
spec:
  value: <custom_claim_value>
```

spec.value フィールドの最大長は 1024 です。ClusterClaim を作成するには **clusterclaims.cluster.open-cluster-management.io** リソースの **create** 権限が必要です。

1.5.14. ManagedClusterSets

ManagedClusterSet は、マネージドクラスターのグループです。マネージドクラスターセット。すべてのマネージドクラスターへのアクセスを管理するのに役立ちます。**ManagedClusterSetBinding** リソースを作成して **ManagedClusterSet** リソースを namespace にバインドすることもできます。

各クラスターは、マネージドクラスターセットのメンバーである必要があります。ハブクラスターをインストールすると、**default** という名前の **ManagedClusterSet** リソースが作成されます。マネージドクラスターセットに割り当てられていないすべてのクラスターは、**default** マネージドクラスターセットに自動的に割り当てられます。**default** マネージドクラスターセットを削除または更新することはできません。

マネージドクラスターセットの作成および管理方法の詳細は、以下を参照してください。

- ManagedClusterSet の作成
- ManagedClusterSetsへの RBAC 権限の割り当て
- ManagedClusterSetBinding リソースの作成
- ManagedClusterSet からのクラスターの削除

1.5.14.1. ManagedClusterSet の作成

マネージドクラスターセットにマネージドクラスターをグループ化して、マネージドクラスターでのユーザーのアクセス権限を制限できます。

必要なアクセス権限: クラスターの管理者

ManagedClusterSet は、クラスタースコープのリソースであるため、**ManagedClusterSet** の作成先となるクラスターで管理者権限が必要です。マネージドクラスターは、複数の **ManagedClusterSet** に追加できません。マネージドクラスターセットは、multicluster engine Operator コンソールまたは CLI から作成できます。

注記: マネージドクラスターセットに追加されていないクラスタープールは、デフォルトの **ManagedClusterSet** リソースに追加されません。クラスターがクラスタープールから要求されると、クラスターはデフォルトの **ManagedClusterSet** に追加されます。

マネージドクラスターを作成すると、管理を容易にするために次のものが自動的に作成されます。

- **global** と呼ばれる **ManagedClusterSet**。
- **open-cluster-management-global-set** という namespace。
- **global** と呼ばれる **ManagedClusterSetBinding** は、**global ManagedClusterSet** を **open-cluster-management-global-set** namespace にバインドします。
  重要: global マネージドクラスターセットを削除、更新、または編集することはできません。**global** マネージドクラスターセットには、すべてのマネージドクラスターが含まれます。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  name: global
  namespace: open-cluster-management-global-set
spec:
  clusterSet: global
```

1.5.14.1.1. CLI を使用した ManagedClusterSet の作成

CLI を使用してマネージドクラスターセットの定義を YAML ファイルに追加し、マネージドクラスターセットを作成します。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: <cluster_set>
```

<cluster_set> をマネージドクラスターセットの名前に置き換えます。

1.5.14.1.2. クラスターの ManagedClusterSet への追加

ManagedClusterSet の作成後に、コンソールまたは CLI を使用してクラスターをマネージドクラスターセットに追加できます。

1.5.14.1.3. CLI を使用したクラスターの ManagedClusterSet への追加

CLI を使用してクラスターをマネージドクラスターに追加するには、以下の手順を実行します。

1. **managedclustersets/join** の仮想サブリソースに作成できるように、RBAC **ClusterRole** エントリーが追加されていることを確認します。
   注記: この権限がないと、マネージドクラスターを **ManagedClusterSet** に割り当てることはできません。このエントリーが存在しない場合は、YAML ファイルに追加します。以下の例を参照してください。

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: clusterrole1
rules:
  - apiGroups: ["cluster.open-cluster-management.io"]
    resources: ["managedclustersets/join"]
    resourceNames: ["<cluster_set>"]
    verbs: ["create"]
```

   <cluster_set> を **ManagedClusterSet** の名前に置き換えます。

   注記: マネージドクラスターを別の **ManagedClusterSet** に移動する場合には、両方のマネージドクラスターセットで権限の設定が必要です。

2. YAML ファイルでマネージドクラスターの定義を検索します。以下の定義例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: <cluster_name>
spec:
  hubAcceptsClient: true
```

3. **cluster.open-cluster-management.io/clusterset** パラメーターを追加し、**ManagedClusterSet** の名前を指定します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: <cluster_name>
  labels:
    cluster.open-cluster-management.io/clusterset: <cluster_set>
spec:
  hubAcceptsClient: true
```

1.5.14.2. ManagedClusterSetへの RBAC 権限の割り当て

ハブクラスターに設定したアイデンティティープロバイダーが提供するクラスターセットに、ユーザーまたはグループを割り当てることができます。

必要なアクセス権限: クラスターの管理者

ManagedClusterSet API RBAC 権限レベルは、以下の表を参照してください。

クラスターセット	アクセス権限	権限の作成
admin	マネージドクラスターセットに割り当てられたすべてのクラスターおよびクラスタープールリソースへのフルアクセス権限。	クラスターの作成、クラスターのインポート、クラスタープールの作成権限。権限は、作成時にマネージドクラスターセットに割り当てる必要があります。
bind	**ManagedClusterSetBinding** を作成してクラスターセットを namespace にバインドするアクセス許可。ユーザーまたはグループには、ターゲット namespace で **ManagedClusterSetBinding** を作成する権限も必要です。マネージドクラスターセットに割り当てられた、すべてのクラスターおよびクラスタープールリソースに対する読み取り専用の権限。	クラスターの作成、クラスターのインポート、またはクラスタープールの作成を実行する権限なし。
view	マネージドクラスターセットに割り当てられたすべてのクラスターおよびクラスタープールリソースに対する読み取り専用権限。	クラスターの作成、クラスターのインポート、またはクラスタープールの作成を実行する権限なし。

注意: グローバルクラスターセットにクラスターセット **admin** 権限を適用することはできません。

コンソールからマネージドクラスターセットにユーザーまたはグループを割り当てるには、次の手順を実行します。

1. OpenShift Container Platform コンソールから、**Infrastructure** > **Clusters** に移動します。
2. **Cluster sets** タブを選択します。
3. ターゲットクラスターセットを選択します。
4. **Access management** タブを選択します。
5. **Add user or group** を選択します。
6. アクセス権を割り当てるユーザーまたはグループを検索して選択します。
7. **Cluster set admin** または **Cluster set view** ロールを選択して、選択したユーザーまたはグループに付与します。詳細は、multicluster engine operator のロールベースのアクセス制御 の **ロールの概要** を参照してください。
8. **Add** を選択して変更を送信します。

テーブルにユーザーまたはグループが表示されます。全マネージドクラスターセットリソースの権限の割り当てがユーザーまたはグループに伝播されるまでに数秒かかる場合があります。

placement 情報は、ManagedCusterSets からの ManagedClusters のフィルタリング を参照してください。

1.5.14.3. ManagedClusterSetBinding リソースの作成

ManagedClusterSetBinding リソースは、**ManagedClusterSet** リソースを namespace にバインドします。同じ namespace で作成されたアプリケーションおよびポリシーは、バインドされたマネージドクラスターセットリソースに含まれるクラスターにのみアクセスできます。

namespace へのアクセス権限は、その namespace にバインドされるマネージドクラスターセットに自動的に適用されます。その namespace へのアクセス権限を持つ場合、その namespace にバインドされるマネージドクラスターセットへのアクセス権限が自動的に付与されます。マネージドクラスターセットにアクセスする権限のみがある場合、namespace の他のマネージドクラスターセットにアクセスする権限は自動的に割り当てられません。

コンソールまたはコマンドラインを使用してマネージドクラスターセットバインドを作成できます。

1.5.14.3.1. コンソールを使用した ManagedClusterSetBinding の作成

コンソールを使用して **ManagedClusterSetBinding** を作成するには、以下の手順を実行します。

1. OpenShift Container Platform コンソールから、**Infrastructure** > **Clusters** に移動し、**Cluster sets** タブを選択します。
2. バインドを作成するクラスターセットの名前を選択します。
3. **Actions** > **Edit namespace bindings** に移動します。
4. **Edit namespace bindings** ページで、ドロップダウンメニューからクラスターセットをバインドする namespace を選択します。

1.5.14.3.2. CLI を使用した ManagedClusterSetBinding の作成

CLI を使用して **ManagedClusterSetBinding** を作成するには、以下の手順を実行します。

1. YAML ファイルに **ManagedClusterSetBinding** リソースを作成します。
   注記: マネージドクラスターセットバインドを作成する場合、マネージドクラスターセットバインドの名前は、バインドするマネージドクラスターセットの名前と一致する必要があります。**ManagedClusterSetBinding** リソースは、以下の情報のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSetBinding
metadata:
  namespace: <namespace>
  name: <cluster_name>
spec:
  clusterSet: <cluster_set>
```

2. ターゲットのマネージドクラスターセットでのバインド権限を割り当てておく必要があります。次の **ClusterRole** リソースの例を表示します。これには、ユーザーが **<cluster_set>** にバインドすることを許可するルールが含まれています。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <clusterrole>
rules:
  - apiGroups: ["cluster.open-cluster-management.io"]
```

```
resources: ["managedclustersets/bind"]
resourceNames: ["<cluster_set>"]
verbs: ["create"]
```

1.5.14.4. Taint と Toleration を使用したマネージドクラスターの配置

Taint と Toleration を使用して、マネージドクラスターまたはマネージドクラスターセットの placement を制御できます。Taint と Toleration は、特定の placement でマネージドクラスターが選択されないようにする方法を提供します。この制御は、特定のマネージドクラスターが一部の placement に含まれないようにする場合に役立ちます。Taint をマネージドクラスターに、Toleration を placement に追加できます。Taint と Toleration が一致しない場合は、マネージドクラスターはその placement には選択されません。

1.5.14.4.1. マネージドクラスターへの Taint の追加

Taint はマネージドクラスターのプロパティーで指定され、 placement がマネージドクラスターまたはマネージドクラスターのセットを除外できます。以下の例のようなコマンドを入力して、Taint をマネージドクラスターに追加できます。

```
oc taint ManagedCluster <managed_cluster_name> key=value:NoSelect
```

Taint の仕様には以下のフィールドが含まれます。

- **(必須)** key: クラスターに適用される Taint キー。この値は、その placement に追加される基準を満たすように、マネージドクラスターの Toleration の値と一致させる必要があります。この値は確認できます。たとえば、この値は **bar** または **foo.example.com/bar** です。
- **(オプション)** Value: Taint キーのTaint値。この値は、その placement に追加される基準を満たすように、マネージドクラスターの Toleration の値と一致させる必要があります。たとえば、この値は **value** とすることができます。
- **(必須)** Effect: Taint を許容しない placement における Taint の効果、または placement の Taint と Toleration が一致しないときに何が起こるか。effect の値は、以下のいずれかの値である必要があります。
  - **NoSelect**: placement は、この Taint を許容しない限り、クラスターを選択できません。Taint の設定前に placement でクラスターが選択された場合は、クラスターは placement の決定から削除されます。
  - **NoSelectIfNew**: スケジューラーは、クラスターが新しい場合にそのクラスターを選択できません。プレイスメントは、Taint を許容し、すでにクラスター決定にそのクラスターがある場合にのみ、そのクラスターを選択することができます。
- **(必須) TimeAdded**: Taint を追加した時間。この値は自動的に設定されます。

1.5.14.4.2. マネージドクラスターのステータスを反映させる組み込み Taint の特定

マネージドクラスターにアクセスできない場合には、クラスターを placement に追加しないでください。以下の Taint は、アクセスできないマネージドクラスターに自動的に追加されます。

- **cluster.open-cluster-management.io/unavailable**: この Taint は、ステータスが **False** の **ManagedClusterConditionAvailable** の条件がある場合にマネージドクラスターに追加されます。Taint には **NoSelect** と同じ効果があり、空の値を指定すると、利用不可のクラスターがスケジュールされないようにできます。この Taint の例は、以下の内容のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
 name: cluster1
spec:
 hubAcceptsClient: true
 taints:
   - effect: NoSelect
     key: cluster.open-cluster-management.io/unavailable
     timeAdded: '2022-02-21T08:11:54Z'
```

- **cluster.open-cluster-management.io/unreachable** - **ManagedClusterConditionAvailable** の条件のステータスが **Unknown** であるか、条件がない場合に、この Taint はマネージドクラスターに追加されます。この Taint には **NoSelect** と同じ効果があり、空の値を指定すると、到達不可のクラスターがスケジュールされないようにできます。この Taint の例は、以下の内容のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'
```

1.5.14.4.3. Toleration の placement への追加

Toleration は placement に適用され、 placement の Toleration と Taint が同じでないマネージドクラスターを placement から除外できます。Toleration の仕様には以下のフィールドが含まれます。

- **(任意)** Key: キーは placement ができるように Taint キーに一致します。
- **(任意)** Value: toleration の値は、 placement を許可する Toleration の Taint の値と一致する必要があります。
- **(任意)** Operator: 演算子はキーと値の関係を表します。有効な演算子は **equal** と **exists** です。デフォルト値は **equal** です。Toleration は、キーが同じ場合、効果が同じ場合、さらび演算子が以下の値のいずれかである場合に、Taint にマッチします。
  - **equal**: Operator は **equal** で、値は Taint および Toleration と同じになります。
  - **exists**: 値のワイルドカード。これにより、 placement は特定のカテゴリーのすべての Taint を許容できます。
- **(任意)** Effect: 一致する Taint の効果。空のままにすると、すべての Taint の効果と一致します。指定可能な値は、**NoSelect** または **NoSelectIfNew** です。
- **(任意)** TolerationSeconds: マネージドクラスターを新しい placement に移動する前に、Taint を許容する時間の長さ (秒単位) です。effect 値が **NoSelect** または **PreferNoSelect** でない場合は、このフィールドは無視されます。デフォルト値は **nil** で、時間制限がないことを示しま

す。**TolerationSeconds** のカウント開始時刻は、クラスターのスケジュール時刻や **TolerationSeconds** 加算時刻の値ではなく、自動的に Taint の **TimeAdded** の値として記載されます。

以下の例は、Taint が含まれるクラスターを許容する Toleration を設定する方法を示しています。

- この例のマネージドクラスターの Taint:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: gpu
      value: "true"
      timeAdded: '2022-02-21T08:11:06Z'
```

- Taint を許容できる placement の Toleration

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal
```

Toleration の定義例では、**key: gpu** と **value: "true"** が一致するため、placement で **cluster1** を選択できます。

注記: マネージドクラスターは、Taint の Toleration が含まれる placement に置かれる保証はありません。他の placement に同じ Toleration が含まれる場合には、マネージドクラスターはそれらの placement のいずれかに置かれる可能性があります。

1.5.14.4.4. 一時的な Toleration の指定

TolerationSeconds の値は、Toleration が Taint を許容する期間を指定します。この一時的な Toleration は、マネージドクラスターがオフラインで、このクラスターにデプロイされているアプリケーションを、許容時間中に別のマネージドクラスターに転送できる場合に役立ちます。

たとえば、以下の Taint を持つマネージドクラスターに到達できなくなります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
```

```
  - effect: NoSelect
    key: cluster.open-cluster-management.io/unreachable
    timeAdded: '2022-02-21T08:11:06Z'
```

以下の例のように、**TolerationSeconds** の値で placement を定義すると、ワークロードは 5 分後に利用可能な別のマネージドクラスターに転送されます。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: demo4
  namespace: demo1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
      tolerationSeconds: 300
```

マネージドクラスターに到達できなくなると、アプリケーションが 5 分間別のマネージドクラスターに移動されます。

1.5.14.5. ManagedClusterSet からのマネージドクラスターの削除

マネージドクラスターセットからマネージドクラスターを削除して別のマネージドクラスターセットに移動するか、セットの管理設定から削除する必要がある場合があります。コンソールまたは CLI を使用して、マネージドクラスターセットからマネージドクラスターを削除できます。

注記:

- マネージドクラスターはすべて、マネージドクラスターセットに割り当てる必要があります。**ManagedClusterSet** からマネージドクラスターを削除し、別の **ManagedClusterSet** に割り当てない場合は、そのクラスターは **default** のマネージドクラスターセットに自動的に追加されます。
- Submariner アドオンがマネージドクラスターにインストールされている場合は、アドオンをアンインストールしてから、マネージドクラスターを **ManagedClusterSet** から削除する必要があります。

1.5.14.5.1. コンソールを使用した ManagedClusterSet からのクラスターの削除

コンソールを使用してマネージドクラスターセットからクラスターを削除するには、次の手順を実行します。

1. **Infrastructure** > **Clusters** をクリックし、**Cluster sets** タブが選択されていることを確認します。
2. マネージドクラスターセットから削除するクラスターセットの名前を選択し、クラスターセットの詳細を表示します。
3. **Actions** > **Manage resource assignments** を選択します。
4. **Manage resource assignments** ページで、クラスターセットから削除するリソースのチェックボックスをオフにします。
   この手順では、すでにクラスターセットのメンバーであるリソースを削除します。マネージドクラスターの詳細を表示して、リソースがすでにクラスターセットのメンバーであるかどうかを確認できます。

注記: マネージドクラスターを別のマネージドクラスターセットに移動する場合には、マネージドクラスターセット両方で RBAC 権限の設定が必要です。

1.5.14.5.2. CLI を使用した ManagedClusterSet からのクラスターの削除

コマンドラインを使用してマネージドクラスターセットからクラスターを削除するには、以下の手順を実行します。

1. 以下のコマンドを実行して、マネージドクラスターセットでマネージドクラスターのリストを表示します。

```
oc get managedclusters -l cluster.open-cluster-management.io/clusterset=<cluster_set>
```

   <cluster_set> をマネージドクラスターセットの名前に置き換えます。

2. 削除するクラスターのエントリーを見つけます。
3. 削除するクラスターの YAML エントリーからラベルを削除します。ラベルの例については、以下のコードを参照してください。

```
labels:
  cluster.open-cluster-management.io/clusterset: clusterset1
```

注記: マネージドクラスターを別のクラスターセットに移動する場合は、マネージドクラスターセット両方で RBAC 権限の設定が必要です。

1.5.15. Placement

placement リソースは、placement namespace にバインドされている **ManagedClusterSets** から **ManagedClusters** のセットを選択するルールを定義する、namespace スコープのリソースです。

必要なアクセス権: クラスター管理者、クラスターセット管理者

プレースメントの使用方法の詳細については、読み続けてください。

- Placement の概要
- ManagedCusterSet からの ManagedClusters のフィルタリング
- PlacementDecisions を使用して選択した ManagedClusters を確認する

1.5.15.1. Placement の概要

マネージドクラスターを使用した配置がどのように機能するかについては、次の情報を参照してください。

- Kubernetes クラスターは、cluster スコープの **ManagedClusters** としてハブクラスターに登録されます。
- **managedcluster** は、クラスタースコープの **ManagedClusterSets** に編成されます。
- **ManagedClusterSets** はワークロード namespace にバインドされます。
- namespace スコープの placement では、潜在的な **ManagedClusters** の作業セットを選択する **ManagedClusterSets** の一部を指定します。

- placement は、**labelSelector** と **claimSelector** を使用して、**ManagedClusterSets** から **ManagedClusters** をフィルター処理します。
- **ManagedClusters** の placement は、Taint と Toleration を使用して制御できます。
- Placements は、要件によってクラスターをランク付けし、そこからクラスターのサブセットを選択します。

注記:

- namespace に **ManagedClusterSetBinding** を作成して、その namespace に **ManagedClusterSet** を最低でも 1 つバインドする必要があります。
- **managedclustersets/bind** の仮想サブリソースの **CREATE** に対してロールベースのアクセスが必要です。

1.5.15.1.1. 関連情報

- 詳細は、Taint と Toleration を使用したマネージドクラスターの配置 を参照してください。
- API の詳細は、Placements API を参照してください。
- placement を伴う ManagedClusters の選択 に戻ります。

1.5.15.2. ManagedClusterSets からの ManagedClusters のフィルタリング

labelSelector または **claimSelector** を使用して、フィルタリングする **ManagedClusters** を選択できます。両方のフィルターの使用方法については、次の例を参照してください。

- 次の例では、**labelSelector** はラベル **vendor: OpenShift** を持つクラスターのみを照合します。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
```

- 次の例では、**claimSelector** は、**region.open-cluster-management.io** と **us-west-1** を持つクラスターのみを照合します。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        claimSelector:
```

```
      matchExpressions:
        - key: region.open-cluster-management.io
          operator: In
          values:
            - us-west-1
```

- また、**clusterSets** パラメーターを使用して、特定のクラスターセットから **ManagedClusters** を フィルター処理することもできます。次の例では、**claimSelector** はクラスターセット **clusterset1** および **clusterset2** のみに一致します。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  clusterSets:
    - clusterset1
    - clusterset2
  predicates:
    - requiredClusterSelector:
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

また、**numberOfClusters** パラメーターを使用して、フィルタリングする **ManagedClusters** の数を選択することもできます。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 3 (1)
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

(1) 選択する **ManagedClusters** の数を指定します。前の例では **3** に設定されています。

1.5.15.2.1. Placement を使用して許容範囲を定義することによる ManagedClusters のフィルタリング

一致するテイントを使用して **ManagedClusters** をフィルタリングする方法は、次の例を参照してください。

- デフォルトでは、次の例では、Placement で **クラスター 1** を選択できません。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: gpu
      value: "true"
      timeAdded: '2022-02-21T08:11:06Z'
```

cluster1 を選択するには、許容範囲を定義する必要があります。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal
```

tolerationSeconds パラメーターを使用して、指定した期間、一致するテイントを持つ **ManagedClusters** を選択することもできます。**tolerationSeconds** は、許容がテイントにバインドされ続ける期間を定義します。**tolerationSeconds** は、指定された時間が経過すると、オフラインになったクラスターにデプロイされたアプリケーションを別のマネージドクラスターに自動的に転送できます。

次の例を見て、**tolerationSeconds** の使用方法を学習します。

- 次の例では、マネージドクラスターにアクセスできなくなります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  name: cluster1
spec:
  hubAcceptsClient: true
  taints:
    - effect: NoSelect
      key: cluster.open-cluster-management.io/unreachable
      timeAdded: '2022-02-21T08:11:06Z'
```

tolerationSeconds を使用して配置を定義すると、ワークロードは別の使用可能なマネージドクラスターに転送されます。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
      tolerationSeconds: 300 (1)
```

(1) 何秒後にワークロードを転送するかを指定します。

1.5.15.2.2. 配置を使用して priorityrPolicy を定義することによる ManagedClusters の優先順位付け

次の例を参照して、**prioritorerPolicy** パラメーターと配置を使用して **ManagedClusters** に優先順位を付ける方法を学習します。

- 次の例では、割り当て可能なメモリーが最大のクラスターを選択します。
  注記: Kubernetes **Node Allocatable** と同様に、allocatable は、各クラスターの Pod で利用可能なコンピュートリソースの量として定義されます。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
```

- 次の例では、割り当て可能な最大の CPU とメモリーを持つクラスターを選択し、リソースの変更に敏感な配置を行います。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableCPU
        weight: 2
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
        weight: 2
```

- 次の例では、**addOn** スコアの CPU 比率が最も大きい 2 つのクラスターを選択し、配置の決定を固定します。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement
  namespace: ns1
spec:
  numberOfClusters: 2
  prioritizerPolicy:
    mode: Exact
    configurations:
      - scoreCoordinate:
          builtIn: Steady
        weight: 3
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: cpuratio
```

1.5.15.2.3. アドオンのステータスに基づいた ManagedClusters のフィルタリング

デプロイされているアドオンのステータスに基づいて、プレースメント用のマネージドクラスターを選択することもできます。たとえば、マネージドクラスターで有効になっている特定のアドオンがある場合にのみ、placement にマネージドクラスターを選択できます。

placement を作成するときに、アドオンのラベルとそのステータスを指定できます。マネージドクラスターでアドオンが有効になっている場合、ラベルは **ManagedCluster** リソース上に自動的に作成されます。アドオンが無効になると、ラベルは自動的に削除されます。

各アドオンは、**feature.open-cluster-management.io/addon-<addon_name>=<status_of_addon>** の形式でラベルで表現します。

addon_name を選択したマネージドクラスターで有効にするアドオンの名前に置き換えます。

status_of_addon をマネージドクラスターが選択されている場合にアドオンに設定するステータスに置き換えます。

status_of_addon に指定できる値については、次の表を参照してください。

値	説明
available	アドオンは有効化されており、利用可能です。
unhealthy	アドオンは有効ですが、リースは継続的に更新されません。
unreachable	アドオンは有効ですが、そのアドオンのリースが見つかりません。これは、マネージドクラスターがオフライン時にも発生する可能性があります。

たとえば、使用可能な **application-manager.** アドオンは、マネージドクラスター上の次のようなラベルで表されます。

```
feature.open-cluster-management.io/addon-application-manager: available
```

アドオンとそのステータスに基づいて placement を作成する方法については、次の例を参照してください。

- 次の placement 例には、**application-manager** が有効になっているすべてのマネージドクラスターが含まれています。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchExpressions:
            - key: feature.open-cluster-management.io/addon-application-manager
              operator: Exists
```

- 次の placement 例には、**application-manager** が **available** ステータスで有効になっているすべてのマネージドクラスターが含まれています。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement2
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            "feature.open-cluster-management.io/addon-application-manager": "available"
```

- 次の placement 例には、**アプリケーションマネージャーが** 無効になっているすべてのマネージドクラスターが含まれています。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement3
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchExpressions:
            - key: feature.open-cluster-management.io/addon-application-manager
              operator: DoesNotExist
```

1.5.15.2.4. 関連情報

- 詳細は、ノード割り当て可能 を参照してください。
- 他のトピックは Selecting ManagedClusters with placement に戻ります。

1.5.15.3. PlacementDecisions を使用した選択した ManagedClusters の確認

ラベル **cluster.open-cluster-management.io/placement={placement_name}** を持つ 1 つ以上の **PlacementDecision** 種類が、プレースメントによって選択された **ManagedClusters** を表すために作成されます。

ManagedCluster が選択され、**PlacementDecision** に追加された場合、この配置を使用するコンポーネントがこの **ManagedCluster** にワークロードを適用する可能性があります。**ManagedCluster** が選択されなくなり、**PlacementDecision** から削除されると、この **ManagedCluster** に適用されているワークロードが削除されます。API の詳細は、**PlacementDecisions API** を参照してください。

次の **PlacementDecision** の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: PlacementDecision
metadata:
  labels:
    cluster.open-cluster-management.io/placement: placement1
  name: placement1-kbc7q
  namespace: ns1
  ownerReferences:
    - apiVersion: cluster.open-cluster-management.io/v1beta1
      blockOwnerDeletion: true
      controller: true
      kind: Placement
      name: placement1
      uid: 05441cf6-2543-4ecc-8389-1079b42fe63e
status:
  decisions:
    - clusterName: cluster1
      reason: ''
    - clusterName: cluster2
      reason: ''
    - clusterName: cluster3
      reason: ''
```

1.5.15.3.1. 関連情報

- 詳細は、PlacementDecisions API を参照してください。

1.5.16. クラスタープールの管理 (テクノロジープレビュー)

クラスタープールは、Red Hat OpenShift Container Platform クラスターにオンデマンドで、スケーリングする場合に、迅速かつコスト効果を高く保ちながら、アクセスできるようにします。クラスタープールは、Amazon Web Services、Google Cloud Platform または Microsoft Azure で設定可能な数多くの OpenShift Container Platform クラスターをプロビジョニングします。このプールは、開発、継続統合、および実稼働のシナリオにおいてクラスター環境を提供したり、置き換えたりする場合に特に便利です。実行を継続するクラスターの数を指定して、すぐに要求できるようにすることができます。残りのクラスターは休止状態に保たれるため、数分以内に再開して要求できます。

ClusterClaim リソースは、クラスタープールからクラスターをチェックアウトするために使用されます。クラスター要求が作成されると、その要求にプールは実行中のクラスターを割り当てます。実行中のクラスターがない場合は、休止状態のクラスターを再開してクラスターを提供するか、新規クラスターをプロビジョニングします。クラスタープールは自動的に新しいクラスターを作成し、休止状態のクラスターを再開して、プール内で利用可能な実行中のクラスターの指定サイズおよび数を維持します。

- クラスタープールの作成
- クラスタープールからのクラスターの要求
- クラスタープールリリースイメージの更新
- クラスタープールのスケーリング
- クラスタープールの破棄

クラスタープールを作成する手順は、クラスターの作成手順と似ています。クラスタープールのクラスターは、すぐ使用するために作成されるわけではありません。

1.5.16.1. クラスタープールの作成

クラスタープールを作成する手順は、クラスターの作成手順と似ています。クラスタープールのクラスターは、すぐ使用するために作成されるわけではありません。

必要なアクセス権限: 管理者

1.5.16.1.1. 前提条件

クラスタープールを作成する前に、次の前提条件を参照してください。

- multicluster engine Operator ハブクラスターをデプロイする必要があります。
- プロバイダー環境で Kubernetes クラスターを作成できるように、multicluster engine Operator ハブクラスターのインターネットアクセスが必要です。
- AWS、GCP、または Microsoft Azure プロバイダーのクレデンシャルが必要です。詳細は、認証情報の管理の概要 を参照してください。
- プロバイダー環境で設定済みのドメインが必要です。ドメインの設定方法は、プロバイダーのドキュメントを参照してください。
- プロバイダーのログイン認証情報が必要です。
- OpenShift Container Platform イメージプルシークレットが必要です。イメージプルシークレットの使用 を参照してください。

注意: この手順でクラスタープールを追加すると、プールからクラスターを要求するときに、multicluster engine Operator によって管理されるクラスターが自動的にインポートされるように設定されます。クラスター要求で管理用に、要求されたクラスターを自動的にインポートしないようにクラスタープールを作成する場合には、**clusterClaim** 理 s-すを以下ののテーションに追加します。

```
kind: ClusterClaim
metadata:
  annotations:
    cluster.open-cluster-management.io/createmanagedcluster: "false" (1)
```

1 文字列であることを示すには、**"false"** という単語を引用符で囲む必要があります。

1.5.16.1.2. クラスタープールを作成する

クラスタープールを作成するには、ナビゲーションメニューで **Infrastructure** > **Clusters** を選択します。**Cluster pools** タブには、アクセス可能なクラスタープールがリスト表示されます。**Create cluster pool** を選択し、コンソールの手順を実行します。

クラスタープールに使用するインフラストラクチャー認証情報がない場合は、**Add credential** を選択して作成できます。

リストから既存の namespace を選択するか、作成する新規 namespace の名前を入力します。クラスタープールは、クラスターと同じ namespace に配置する必要はありません。

クラスタープールの RBAC ロールを使用して、既存クラスターセットのロール割り当てを共有する場合は、クラスターセット名を選択します。クラスタープールのクラスターセットは、クラスタープールの作成時にのみ設定できます。クラスタープールの作成後には、クラスタープールまたはクラスタープールのクラスターセットの関連付けを変更できません。クラスタープールから要求したクラスターは、クラスタープールと同じクラスターセットに自動的に追加されます。

注記: cluster admin の権限がない場合は、クラスターセットを選択する必要があります。この状況でクラスターセットの名前が含まれない場合は、禁止エラーで、クラスターセットの作成要求が拒否されます。選択できるクラスターセットがない場合は、クラスター管理者に連絡してクラスターセットを作成し、**clusterset admin** 権限を付与してもらいます。

cluster pool size は、クラスタープールにプロビジョニングするクラスターの数を指定し、クラスタープールの実行回数は、プールが実行を継続し、すぐに使用できるように要求できるクラスターの数を指定します。

この手順は、クラスターを作成する手順と非常に似ています。

プロバイダーに必要な固有の情報は、以下を参照してください。

- Amazon Web Services でのクラスターの作成
- Google Cloud Platform でのクラスターの作成
- Microsoft Azure でのクラスターの作成

1.5.16.2. クラスタープールからのクラスターの要求

ClusterClaim リソースは、クラスタープールからクラスターをチェックアウトするために使用されます。クラスターの稼働中で、クラスタープールで準備できると、要求が完了します。クラスタープールは、クラスタープールに指定された要件を維持するために、クラスタープールに新しい実行中およびハイバネートされたクラスターを自動的に作成します。

注記: クラスタープールから要求されたクラスターが不要になり、破棄されると、リソースは削除されます。クラスターはクラスタープールに戻りません。

必要なアクセス権限: 管理者

1.5.16.2.1. 前提条件

クラスタープールからクラスターを要求する前に、以下を利用意する必要があります。

利用可能なクラスターのある/ないクラスタープール。クラスタープールに利用可能なクラスターがある場合、利用可能なクラスターが要求されます。クラスタープールに利用可能なクラスターがない場合は、要求を満たすためにクラスターが作成されます。クラスタープールの作成方法については、クラスタープールの作成 を参照してください。

1.5.16.2.2. クラスタープールからのクラスターの要求

クラスター要求の作成時に、クラスタープールから新規クラスターを要求します。クラスターが利用可能になると、クラスターはプールからチェックアウトされます。自動インポートを無効にしていない限り、要求されたクラスターはマネージドクラスターの 1 つとして自動的にインポートされます。

以下の手順を実行してクラスターを要求します。

1. ナビゲーションメニューから **Infrastructure** > **Clusters** をクリックし、**Cluster pools** タブを選択します。
2. クラスターを要求するクラスタープールの名前を見つけ、**Claim cluster** を選択します。

クラスターが利用可能な場合には、クラスターが要求され、**マネージドクラスター** タブにすぐに表示されます。利用可能なクラスターがない場合は、休止状態のクラスターの再開や、新しいクラスターのプロビジョニングに数分かかる場合があります。この間、要求のステータスは **pending** です。クラスタープールをデプロイメントして、保留中の要求を表示または削除します。

要求されたクラスターは、クラスタープールにあった時に関連付けられたクラスターセットに所属します。要求時には、要求したクラスターのクラスターセットは変更できません。

注記: クラウドプロバイダーの認証情報のプルシークレット、SSH キー、またはベースドメインへの変更は、クラスタープールから請求された既存のクラスターについては、元の認証情報を使用してすでにプロビジョニングされているため、反映されません。コンソールを使用してクラスタープール情報を編集することはできませんが、CLI インターフェイスを使用してその情報を更新することで更新できます。更新された情報を含む認証情報を使用して、新しいクラスタープールを作成することもできます。新しいプールで作成されるクラスターは、新しい認証情報で提供される設定を使用します。

1.5.16.3. クラスタープールリリースイメージの更新

クラスタープールのクラスターが一定期間、休止状態のままになると、クラスターの Red Hat OpenShift Container Platform リリースイメージがバックレベルになる可能性があります。このような場合は、クラスタープールにあるクラスターのリリースイメージのバージョンをアップグレードしてください。

必要なアクセス: 編集

クラスタープールにあるクラスターの OpenShift Container Platform リリースイメージを更新するには、以下の手順を実行します。

注記: この手順では、クラスタープールですでに要求されているクラスタープールからクラスターを更新しません。この手順を完了すると、リリースイメージの更新は、クラスタープールに関連する次のクラスターにのみ適用されます。

- この手順でリリースイメージを更新した後にクラスタープールによって作成されたクラスター。
- クラスタープールで休止状態になっているクラスター。古いリリースイメージを持つ既存の休止状態のクラスターは破棄され、新しいリリースイメージを持つ新しいクラスターがそれらを置き換えます。

1. ナビゲーションメニューから **infrastructure** > **Clusters** をクリックします。
2. **Cluster pools** タブを選択します。
3. **クラスタープール** の表で、更新するクラスタープールの名前を見つけます。
4. 表の **Cluster pools** の **Options** メニューをクリックし、**Update release image**を選択します。
5. このクラスタープールから今後、クラスターの作成に使用する新規リリースイメージを選択します。

クラスタープールのリリースイメージが更新されました。

ヒント: アクション 1 つで複数のクラスターのリリースイメージを更新するには、各クラスタープールのボックスを選択して **Actions** メニューを使用し、選択したクラスタープールのリリースイメージを更新します。

1.5.16.4. Scaling cluster pools (Technology Preview)

クラスタープールのクラスター数は、クラスタープールサイズのクラスター数を増やしたり、減らしたりして変更できます。

必要なアクセス権限: クラスターの管理者

クラスタープールのクラスター数を変更するには、以下の手順を実行します。

1. ナビゲーションメニューから **infrastructure** > **Clusters** をクリックします。
2. **Cluster pools** タブを選択します。
3. 変更するクラスタープールの **Options** メニューで、**Scale cluster pool** を選択します。
4. プールサイズの値を変更します。
5. オプションで、実行中のクラスターの数を更新して、要求時にすぐに利用可能なクラスター数を増減できます。

クラスタープールは、新しい値を反映するようにスケーリングされます。

1.5.16.5. クラスタープールの破棄

クラスタープールを作成し、不要になった場合は、そのクラスタープールを破棄できます。

重要: クラスター要求がないクラスタープールのみ破棄できます。

必要なアクセス権限: クラスターの管理者

クラスタープールを破棄するには、以下の手順を実行します。

1. ナビゲーションメニューから **infrastructure** > **Clusters** をクリックします。
2. **Cluster pools** タブを選択します。
3. 削除するクラスタープールの **Options** メニューで、確認ボックスに **confirm** と入力して **Destroy** を選択します。
   注記:

- クラスタープールにクラスター要求がある場合、**Destroy** ボタンは無効になります。
- クラスタープールを含む namespace は削除されません。namespace を削除すると、これらのクラスターのクラスター要求リソースが同じ namespace で作成されるため、クラスタープールから要求されたクラスターが破棄されます。

ヒント: アクション 1 つで複数のクラスタープールを破棄するには、各クラスタープールのボックスを選択して **Actions** メニューを使用し、選択したクラスタープールを破棄します。

1.5.17. ManagedServiceAccount アドオンの有効化

multicluster engine Operator バージョン 2.5 以降をインストールすると、**ManagedServiceAccount** アドオンがデフォルトで有効になります。ハブクラスターを multicluster engine Operator バージョン 2.4 からアップグレードし、アップグレード前に **ManagedServiceAccount** アドオンを有効にしていなかった場合は、アドオンを手動で有効にする必要があります。

ManagedServiceAccount を使用すると、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。

1.5.17.1. 前提条件

- お使いの環境に Red Hat OpenShift Container Platform バージョン 4.13 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。
- multicluster engine Operator がインストールされている必要がある。

1.5.17.2. ManagedServiceAccount の有効化

ハブクラスターとマネージドクラスターの **ManagedServiceAccount** 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 **ManagedServiceAccount** 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 **ManagedServiceAccount** 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: <managedserviceaccount_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
```

1.5.18. クラスターのライフサイクルの詳細設定

一部のクラスター設定は、インストール中またはインストール後に設定できます。

1.5.18.1. API サーバー証明書のカスタマイズ

マネージドクラスターは、OpenShift Kube API サーバーの外部ロードバランサーとの相互接続を介してハブクラスターと通信します。デフォルトの OpenShift Kube API サーバー証明書は、OpenShift Container Platform のインストール時に内部 Red Hat OpenShift Container Platform クラスター認証局 (CA) によって発行されます。必要に応じて、証明書を追加または変更できます。

API サーバー証明書を変更すると、マネージドクラスターとハブクラスター間の通信に影響を与える可能性があります。製品をインストールする前に名前付き証明書を追加すると、マネージドクラスターがオフライン状態になる可能性がある問題を回避できます。

次のリストには、証明書の更新が必要となる場合の例がいくつか含まれています。

- 外部ロードバランサーのデフォルトの API サーバー証明書を独自の証明書に置き換える必要がある。OpenShift Container Platform ドキュメントの **API サーバー証明書の追加** のガイダンスに従い、ホスト名 **api.<cluster_name>.<base_domain>** の名前付き証明書を追加して、外部ロードバランサーのデフォルトの API サーバー証明書を置き換えることができます。証明書を置き換えると、マネージドクラスターの一部がオフライン状態に移行する可能性があります。証明書のアップグレード後にクラスターがオフライン状態になった場合は、**Troubleshooting imported clusters offline after certificate change** のトラブルシューティング手順に従って問題を解決してください。
  注記: 製品をインストールする前に名前付き証明書を追加すると、クラスターがオフライン状態に移行するのを回避できます。

- 外部ロードバランサーの名前付き証明書の有効期限が切れているため、証明書を置き換える必要がある。中間証明書の数に関係なく、古い証明書と新しい証明書の両方が同じルート CA 証明書を共有する場合は、OpenShift Container Platform ドキュメントの **API サーバー証明書の追加** のガイダンスに従って、新しい証明書の新しいシークレットを作成できます。次に、ホスト名 **api.<cluster_name>.<base_domain>** のサービス証明書参照を **APIServer** カスタムリソース内の新しいシークレットに更新します。それ以外の場合、古い証明書と新しい証明書に異なるルート CA 証明書がある場合は、次の手順を実行して証明書を置き換えます。

  1. 次の例のような **APIServer** カスタムリソースを見つけます。

     ```
     apiVersion: config.openshift.io/v1
     kind: APIServer
     metadata:
       name: cluster
     spec:
       audit:
         profile: Default
       servingCerts:
         namedCertificates:
         - names:
           - api.mycluster.example.com
           servingCertificate:
             name: old-cert-secret
     ```

  2. 次のコマンドを実行して、既存の証明書と新しい証明書の内容を含む新しいシークレットを **openshift-config** namespace に作成します。

     a. 古い証明書を新しい証明書にコピーします。

     ```
     cp old.crt combined.crt
     ```

     b. 新しい証明書の内容を古い証明書のコピーに追加します。

     ```
     cat new.crt >> combined.crt
     ```

     c. 結合した証明書を適用してシークレットを作成します。

     ```
     oc create secret tls combined-certs-secret --cert=combined.crt --key=old.key -n
     openshift-config
     ```

  3. **APIServer** リソースを更新して、結合された証明書を **ServingCertificate** として参照します。

     ```
     apiVersion: config.openshift.io/v1
     ```

```
kind: APIServer
metadata:
  name: cluster
spec:
  audit:
    profile: Default
  servingCerts:
    namedCertificates:
    - names:
      - api.mycluster.example.com
      servingCertificate:
        name: combined-cert-secret
```

4. 約 15 分後、新しい証明書と古い証明書の両方を含む CA バンドルがマネージドクラスターに伝播されます。

5. 次のコマンドを入力して、新しい証明書情報のみを含む **new-cert-secret** という名前の別のシークレットを **openshift-config** namespace に作成します。

```
oc create secret tls new-cert-secret --cert=new.crt --key=new.key -n openshift-config
{code}
```

6. **new-cert-secret** を参照するように **servingCertificate** の名前を変更して、**APIServer** リソースを更新します。リソースは以下の例のようになります。

```
apiVersion: config.openshift.io/v1
kind: APIServer
metadata:
  name: cluster
spec:
  audit:
    profile: Default
  servingCerts:
    namedCertificates:
    - names:
      - api.mycluster.example.com
      servingCertificate:
        name: new-cert-secret
```

約 15 分後、古い証明書が CA バンドルから削除され、変更がマネージドクラスターに自動的に伝播されます。

注記: マネージドクラスターは、ホスト名 **api.<cluster_name>.<base_domain>** を使用してハブクラスターにアクセスする必要があります。他のホスト名で設定された名前付き証明書は使用できません。

1.5.18.2. ハブクラスターとマネージドクラスター間のプロキシーの設定

マネージドクラスターを Kubernetes Operator ハブクラスターのマルチクラスターエンジンに登録するには、マネージドクラスターを multicluster engine Operator ハブクラスターに転送する必要があります。マネージドクラスターが multicluster engine Operator ハブクラスターに直接アクセスできない場合があります。この例では、マネージドクラスターからの通信が HTTP または HTTPS プロキシーサーバー経由で multicluster engine Operator ハブクラスターにアクセスできるようにプロキシー設定を指定します。

たとえは、multicluster engine Operator ハフクラスターはパフリッククラウドにあり、マネージドクラスターはファイアウォールの背後にあるプライベートクラウド環境にあります。プライベートクラウドからの通信は、HTTP または HTTPS プロキシーサーバーのみを経由できます。

1.5.18.2.1. 前提条件

- HTTP トンネルをサポートする HTTP または HTTPS プロキシーサーバーが実行されている。(例: HTTP connect メソッド)
- HTTP または HTTPS プロキシーサーバーに到達できる管理クラスターがあり、プロキシーサーバーは multicluster engine Operator ハブクラスターにアクセスできる。

ハブクラスターとマネージドクラスターとの間でプロキシー設定を指定するには、以下の手順を実行します。

1. プロキシー設定を使用して **KlusterConfig** リソースを作成します。
   a. 以下の HTTP プロキシー設定を参照してください。

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: http-proxy
spec:
  hubKubeAPIServerProxyConfig:
    httpProxy: "http://<username>:<password>@<ip>:<port>"
```

   b. 以下の HTTPS プロキシー設定を参照してください。

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: https-proxy
spec:
  hubKubeAPIServerProxyConfig:
    httpsProxy: "https://<username>:<password>@<ip>:<port>"
    caBundle: <user-ca-bundle>
```

   注: HTTPS プロキシーサーバーを設定する場合は、CA 証明書が必要です。HTTP プロキシーと HTTPS プロキシーの両方が指定されている場合は、HTTPS プロキシーが使用されます。

2. マネージドクラスターの作成時に、**KlusterletConfig** リソースを参照するアノテーションを追加して、**KlusterletConfig** リソースを選択します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    agent.open-cluster-management.io/klusterlet-config: <klusterlet-config-name>
  name:<managed-cluster-name>
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

注: マルチクラスターエンジンの operator コンソールで操作する場合は、YAML ビューを切り替えて **ManagedCluster** リソースにアノテーションを追加する必要がある場合があります。

1.5.18.2.2. ハブクラスターとマネージドクラスター間のプロキシーの無効化

開発が変更された場合は、HTTP または HTTPS プロキシーを無効にする必要がある場合があります。

1. **ManagedCluster** リソースに移動します。
2. アノテーション **agent.open-cluster-management.io/klusterlet-config** を削除します。

1.5.18.2.3. オプション: 特定のノードで実行するように klusterlet を設定する

Red Hat Advanced Cluster Management for Kubernetes を使用してクラスターを作成する場合、マネージドクラスターの **nodeSelector** および **tolerations** アノテーションを設定することで、マネージドクラスター klusterlet を実行するノードを指定できます。これらのオプションを設定するには、次の手順を実行します。

1. コンソールのクラスターページから、更新するマネージドクラスターを選択します。
2. YAML コンテンツを表示するには、YAML スイッチを **On** に設定します。
   注記: YAML エディターは、クラスターをインポートまたは作成するときにのみ使用できます。インポートまたは作成後にマネージドクラスターの YAML 定義を編集するには、OpenShift Container Platform コマンドラインインターフェイスまたは Red Hat Advanced Cluster Management 検索機能を使用する必要があります。
3. **nodeSelector** アノテーションをマネージドクラスターの YAML 定義に追加します。このアノテーションのキーは、**open-cluster-management/nodeSelector** です。このアノテーションの値は、JSON 形式の文字列マップです。
4. **tolerations** エントリーをマネージドクラスターの YAML 定義に追加します。このアノテーションのキーは、**open-cluster-management/tolerations** です。このアノテーションの値は、JSON 形式の toleration リストを表します。結果の YAML は次の例のようになります。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    open-cluster-management/nodeSelector: '{"dedicated":"acm"}'
    open-cluster-management/tolerations:
'[{"key":"dedicated","operator":"Equal","value":"acm","effect":"NoSchedule"}]'
```

1.5.18.3. マネージドクラスターをインポートする際のハブクラスター API サーバーのサーバー URL と CA バンドルのカスタマイズ (テクノロジープレビュー)

マネージドクラスターとハブクラスターの間に中間コンポーネントが存在する場合、multicluster engine Operator ハブクラスターにマネージドクラスターを登録できない可能性があります。中間コンポーネントの例には、仮想 IP、ロードバランサー、リバースプロキシー、API ゲートウェイなどがあります。中間コンポーネントがある場合は、マネージドクラスターをインポートするときに、ハブクラスター API サーバーのカスタムサーバー URL と CA バンドルを使用する必要があります。

1.5.18.3.1. 前提条件

- マネージドクラスターからハブクラスター API サーバーにアクセスできるように、中間コンポーネントを設定する必要があります。

- 中間コンポーネントがマネージドクラスターとハブクラスター API サーバー間の SSL 接続を終端する場合は、SSL 接続をブリッジし、元のリクエストからの認証情報をハブクラスター API サーバーのバックエンドに渡す必要があります。Kubernetes API サーバーのユーザー偽装機能を使用すると、SSL 接続をブリッジできます。
  中間コンポーネントは、元のリクエストからクライアント証明書を抽出し、証明書サブジェクトのコモンネーム (CN) と組織 (O) を偽装ヘッダーとして追加し、変更された偽装リクエストをハブクラスター API サーバーのバックエンドに転送します。

  注記: SSL 接続をブリッジすると、クラスタープロキシーアドオンが機能しなくなります。

1.5.18.3.2. サーバー URL とハブクラスター CA バンドルのカスタマイズ

マネージドクラスターをインポートするときにカスタムハブ API サーバー URL と CA バンドルを使用するには、次の手順を実行します。

1. カスタムのハブクラスター API サーバー URL と CA バンドルを使用して **KlusterConfig** リソースを作成します。以下の例を参照してください。

```
apiVersion: config.open-cluster-management.io/v1alpha1
kind: KlusterletConfig
metadata:
  name: 1
spec:
  hubKubeAPIServerURL: "https://api.example.com:6443" 2
  hubKubeAPIServerCABundle: "LS0tLS1CRU...LS0tCg==" 3
```

1. klusterlet 設定名を追加します。
2. カスタムサーバー URL を追加します。
3. カスタム CA バンドルを追加します。

2. マネージドクラスターを作成するときに、リソースを参照するアノテーションを追加して **KlusterletConfig** リソースを選択します。以下の例を参照してください。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    agent.open-cluster-management.io/klusterlet-config: 1
  name: 2
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

1. klusterlet 設定名を追加します。
2. クラスター名を追加します。

注記: コンソールを使用する場合は、**ManagedCluster** リソースにアノテーションを追加するために、YAML ビューを有効にする必要がある場合があります。

1.5.18.4. 関連情報

- API サーバー証明書の追加
- 証明書を変更した後のインポート済みクラスターのオフラインでのトラブルシューティング
- クラスタープロキシーアドオンのプロキシー設定

1.5.19. マネージメントからのクラスターの削除

multicluster engine operator で作成された OpenShift Container Platform クラスターを管理から削除する場合は、それを **デタッチ** または **破棄** することができます。クラスターをデタッチするとマネージメントから削除されますが、完全には削除されません。管理する場合には、もう一度インポートし直すことができます。このオプションは、クラスターが **Ready** 状態にある場合にだけ利用できます。

次の手順により、次のいずれかの状況でクラスターが管理から削除されます。

- すでにクラスターを削除しており、削除したクラスターを Red Hat Advanced Cluster Management から削除したいと考えています。
- クラスターを管理から削除したいが、クラスターを削除していない。

重要:

- クラスターを破棄すると、マネージメントから削除され、クラスターのコンポーネントが削除されます。
- マネージドクラスターを接続解除または破棄すると、関連する namepsace が自動的に削除されます。この namespace にカスタムリソースを配置しないでください。
  - コンソールを使用したクラスターの削除
  - コマンドラインを使用したクラスターの削除
  - クラスター削除後の残りのリソースの削除
  - クラスターの削除後の etcd データベースのデフラグ

1.5.19.1. コンソールを使用したクラスターの削除

ナビゲーションメニューから、**Infrastructure** > **Clusters** に移動し、管理から削除するクラスターの横にあるオプションメニューから **Destroy cluster** または **Detach cluster** を選択します。

ヒント: 複数のクラスターをデタッチまたは破棄するには、デタッチまたは破棄するクラスターのチェックボックスを選択して、**Detach** または **Destroy** を選択します。

注記: **local-cluster** と呼ばれる管理対象時にハブクラスターをデタッチしようとすると、**disableHubSelfManagement** のデフォルト設定が **false** かどうかを確認してください。この設定が原因で、ハブクラスターはデタッチされると、自身を再インポートして管理し、**MultiClusterHub** コントローラーが調整されます。ハブクラスターがデタッチプロセスを完了して再インポートするのに時間がかかる場合があります。

プロセスが終了するのを待たずにハブクラスターを再インポートするには、以下のコマンドを実行して **multiclusterhub-operator** Pod を再起動して、再インポートの時間を短縮できます。

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

ネットワーク接続時のオンラインインストール で説明されているように、**disableHubSelfManagement** の値を **true** に変更して、自動的にインポートされないようにハブクラスターの値を変更できます。

1.5.19.2. コマンドラインを使用したクラスターの削除

ハブクラスターのコマンドラインを使用してマネージドクラスターをデタッチするには、以下のコマンドを実行します。

```
oc delete managedcluster $CLUSTER_NAME
```

切断後にマネージドクラスターを破棄するには、次のコマンドを実行します。

```
oc delete clusterdeployment <CLUSTER_NAME> -n $CLUSTER_NAME
```

注記:

- マネージドクラスターの破壊を防ぐには、**ClusterDeployment** カスタムリソースで **spec.preserveOnDelete** パラメーターを **true** に設定します。
- **disableHubSelfManagement** のデフォルト設定は **false** です。**false`setting causes the hub cluster, also called `local-cluster** 切り離されたときに再インポートして管理し、**MultiClusterHub** コントローラーを調整します。
  切り離しと再インポートのプロセスには数時間かかる場合があり、ハブクラスターが完了するまでに数時間かかる場合があります。プロセスが終了するのを待たずにハブクラスターを再インポートする場合は、以下のコマンドを実行して **multiclusterhub-operator** Pod を再起動して、再インポートの時間を短縮できます。

  ```
  oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
  ```

  disableHubSelfManagement の値を **true** に指定して、自動的にインポートされないように、ハブクラスターの値を変更できます。ネットワーク接続時のオンラインインストール を参照してください。

1.5.19.3. クラスター削除後の残りのリソースの削除

削除したマネージドクラスターにリソースが残っている場合は、残りのすべてのコンポーネントを削除するための追加の手順が必要になります。これらの追加手順が必要な場合には、以下の例が含まれます。

- マネージドクラスターは、完全に作成される前にデタッチされ、**klusterlet** などのコンポーネントはマネージドクラスターに残ります。
- マネージドクラスターをデタッチする前に、クラスターを管理していたハブが失われたり、破棄されているため、ハブからマネージドクラスターをデタッチする方法はありません。
- マネージドクラスターは、デタッチ時にオンライン状態ではありませんでした。

これらの状況の 1 つがマネージドクラスターのデタッチの試行に該当する場合は、マネージドクラスターから削除できないリソースがいくつかあります。マネージドクラスターをデタッチするには、以下の手順を実行します。

1. **oc** コマンドラインインターフェイスが設定されていることを確認してください。

2. また、マネージドクラスターに **KUBECONFIG** が設定されていることを確認してください。**oc get ns | grep open-cluster-management-agent** を実行すると、2 つの namespace が表示されるはずです。

```
open-cluster-management-agent        Active   10m
open-cluster-management-agent-addon   Active   10m
```

3. 次のコマンドを使用して、**klusterlet** カスタムリソースを削除します。

```
oc get klusterlet | grep klusterlet | awk '{print $1}' | xargs oc patch klusterlet --type=merge -p
'{"metadata":{"finalizers": []}}'
```

4. 次のコマンドを実行して、残りのリソースを削除します。

```
oc delete namespaces open-cluster-management-agent open-cluster-management-agent-
addon --wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc delete crds --
wait=false
oc get crds | grep open-cluster-management.io | awk '{print $1}' | xargs oc patch crds --
type=merge -p '{"metadata":{"finalizers": []}}'
```

5. 次のコマンドを実行して、namespaces と開いているすべてのクラスター管理 **crds** の両方が削除されていることを確認します。

```
oc get crds | grep open-cluster-management.io | awk '{print $1}'
oc get ns | grep open-cluster-management-agent
```

1.5.19.4. クラスターの削除後の etcd データベースのデフラグ

マネージドクラスターが多数ある場合は、ハブクラスターの **etcd** データベースのサイズに影響を与える可能性があります。OpenShift Container Platform 4.8 では、マネージドクラスターを削除すると、ハブクラスターの **etcd** データベースのサイズは自動的に縮小されません。シナリオによっては、**etcd** データベースは領域不足になる可能性があります。**etcdserver: mvcc: database space exceeded** のエラーが表示されます。このエラーを修正するには、データベース履歴を圧縮し、**etcd** データベースのデフラグを実行して **etcd** データベースのサイズを縮小します。

注記: OpenShift Container Platform バージョン 4.9 以降では、etcd Operator はディスクを自動的にデフラグし、**etcd** 履歴を圧縮します。手動による介入は必要ありません。以下の手順は、OpenShift Container Platform 4.8 以前のバージョン向けです。

以下の手順を実行して、ハブクラスターで **etcd** 履歴を圧縮し、ハブクラスターで **etcd** データベースをデフラグします。

1.5.19.4.1. 前提条件

- OpenShift CLI (**oc**) がインストールされている。
- **cluster-admin** 権限を持つユーザーとしてログインしている。

1.5.19.4.2. 手順

1. **etcd** 履歴を圧縮します。
   a. 次に、**etcd** メンバーへのリモートシェルセッションを開きます。

```
$ oc rsh -n openshift-etcd etcd-control-plane-0.example.com etcdctl endpoint status --
cluster -w table
```

b. 以下のコマンドを実行して **etcd** 履歴を圧縮します。

```
sh-4.4#etcdctl compact $(etcdctl endpoint status --write-out="json" |  egrep -o '"revision":
[0-9]*' | egrep -o '[0-9]*' -m1)
```

出力例

```
$ compacted revision 158774421
```

2. Defragmenting **etcd** data で説明されているように、**etcd** データベースを デフラグし、**NOSPACE** アラームを消去します。

1.6. DISCOVERY サービスの概要

OpenShift Cluster Manager で利用可能な OpenShift 4 クラスターを検出できます。検出後に、クラスターをインポートして管理できます。Discovery サービスは、バックエンドおよびコンソールでの用途に Discover Operator を使用します。

OpenShift Cluster Manager 認証情報が必要になります。認証情報を作成する必要がある場合は、Red Hat OpenShift Cluster Manager の認証情報の作成 を参照してください。

必要なアクセス権限: 管理者

- コンソールでの検出の設定
- CLI を使用した検出機能の設定

1.6.1. コンソールでの検出の設定

製品のコンソールを使用して検出を有効にします。

必要なアクセス権: 認証情報が作成された namespace へのアクセス権。

1.6.1.1. 前提条件

- 認証情報が必要です。OpenShift Cluster Manager に接続するには、Red Hat OpenShift Cluster Manager の認証情報の作成 を参照してください。

1.6.1.2. 検出の設定

コンソールで検出を設定し、クラスターを検索します。個別の認証情報を使用して複数の **DiscoveryConfig** リソースを作成できます。コンソールの指示に従います。

1.6.1.3. 検出されたクラスターの表示

認証情報を設定してインポートするクラスターを検出した後に、コンソールで表示できます。

1. **Clusters** > **Discovered clusters**の順にクリックします。
2. 以下の情報が投入された表を確認してください。

- **name** は、OpenShift Cluster Manager で指定された表示名です。クラスターに表示名がない場合は、クラスターコンソール URL をもとに生成された名前が表示されます。OpenShift Cluster Manager でコンソール URL がない場合や手動で変更された場合には、クラスターの外部 ID が表示されます。
- **Namespace** は、認証情報および検出クラスターを作成した namespace です。
- **type** は検出されたクラスターの Red Hat OpenShift タイプです。
- **distribution version** は、検出されたクラスターの Red Hat OpenShift バージョンです。
- **infrastrcture provider** は検出されたクラスターのクラウドプロバイダーです。
- **Last active** は、検出されたクラスターが最後にアクティブであった時間です。
- **Created** は検出クラスターが作成された時間です。
- **Discovered** は検出クラスターが検出された時間です。

3. 表の中にある情報はどれでも検索できます。たとえば、特定の namespace で **Discovered clusters** のみを表示するには、その namespace を検索します。
4. **Import cluster** をクリックすると、マネージドクラスターを作成できます。検出クラスターのインポート を参照してください。

1.6.1.4. 検出クラスターのインポート

クラスターの検出後に、コンソールの **Discovered clusters** に表示されるクラスターをインポートできます。

1.6.1.5. 前提条件

検出の設定に使用した namespace へのアクセス権が必要である。

1.6.1.6. 検出クラスターのインポート

1. 既存の **Clusters** ページに移動し、**Discovered clusters** タブをクリックします。
2. **Discovered Clusters** の表から、インポートするクラスターを見つけます。
3. オプションメニューから **Import cluster** を選択します。
4. 検出クラスターの場合は、本書を使用して手動でインポートしたり、Import cluster を自動的に選択したりできます。
5. 認証情報または Kubeconfig ファイルを使用して自動でインポートするには、コンテンツをコピーして貼り付けます。
6. **Import** をクリックします。

1.6.2. CLI を使用した検出の有効化

CLI を使用して検出を有効にし、Red Hat OpenShift Cluster Manager が入手できるクラスターを見つけます。

必要なアクセス権限: 管理者

1.6.2.1. 前提条件

- Red Hat OpenShift Cluster Manager に接続するための認証情報を作成している。

1.6.2.2. 検出の設定とプロセス

注記: DiscoveryConfig は **discovery** という名前に指定し、選択した **credential** と同じ namespace に作成する必要があります。以下の **DiscoveryConfig** のサンプルを参照してください。

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveryConfig
metadata:
  name: discovery
  namespace: <NAMESPACE_NAME>
spec:
  credential: <SECRET_NAME>
  filters:
    lastActive: 7
    openshiftVersions:
    - "4.13"
```

1. **SECRET_NAME** は、以前に設定した認証情報に置き換えます。
2. **NAMESPACE_NAME** は **SECRET_NAME** の namespace に置き換えます。
3. クラスターの最後のアクティビティー (日数) からの最大時間を入力します。たとえば、**lastActive: 7** では、過去 7 日間にアクティブなクラスターが検出されます。
4. Red Hat OpenShift クラスターのバージョンを入力して、文字列の一覧として検出します。**注記: openshiftVersions** 一覧に含まれるエントリーはすべて、OpenShift のメジャーバージョンとマイナーバージョンを指定します。たとえば、**"4.11"** には OpenShift バージョン **4.11** のすべてのパッチリリース (**4.11.1**、**4.11.2** など) が含まれます。

1.6.2.3. 検出されたクラスターの表示

検出されたクラスターを表示するには、**oc get discoveredclusters -n <namespace>** を実行して、**namespace** は検出認証情報が存在する namespace に置き換えます。

1.6.2.3.1. DiscoveredClusters

オブジェクトは Discovery コントローラーにより作成されます。このような **DiscoveredClusters** は、**DiscoveryConfig discoveredclusters.discovery.open-cluster-management.io** API で指定したフィルターと認証情報を使用して OpenShift Cluster Manager で検出されたクラスターを表します。**name** の値はクラスターの外部 ID です。

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveredCluster
metadata:
  name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
  namespace: <NAMESPACE_NAME>
spec:
  activity_timestamp: "2021-04-19T21:06:14Z"
  cloudProvider: vsphere
  console: https://console-openshift-console.apps.qe1-vmware-pkt.dev02.red-chesterfield.com
  creation_timestamp: "2021-04-19T16:29:53Z"
```

```
  credential:
    apiVersion: v1
    kind: Secret
    name: <SECRET_NAME>
    namespace: <NAMESPACE_NAME>
  display_name: qe1-vmware-pkt.dev02.red-chesterfield.com
  name: fd51aafa-95a8-41f7-a992-6fb95eed3c8e
  openshiftVersion: 4.13
  status: Stale
```

1.7. HOSTED CONTROL PLANE

multicluster engine Operator クラスター管理では、スタンドアロンまたは Hosted control plane の 2 つの異なるコントロールプレーン設定を使用して、OpenShift Container Platform クラスターをデプロイできます。スタンドアロン設定では、専用の仮想マシンまたは物理マシンを使用して、OpenShift Container Platform のコントロールプレーンをホストします。OpenShift Container Platform の Hosted control plane を使用すると、コントロールプレーンごとに専用の物理マシンを必要とせずに、ホスティングクラスター上にコントロールプレーンを Pod として作成できます。

OpenShift Container Platform の Hosted Control Plane は、ベアメタルおよび Red Hat OpenShift Virtualization で、また Amazon Web Services (AWS) の **テクノロジープレビュー** 機能として利用できます。コントロールプレーンは、OpenShift Container Platform バージョン 4.14 以降でホスティングできます。Hosted control plane 機能がデフォルトで有効になりました。

1.7.1. 要件

次の表は、各プラットフォームでサポートされている OpenShift Container Platform のバージョンを示しています。この表では、**ホスティング OpenShift Container Platform バージョンは、** Multi-Cluster Engine Operator が有効になっている OpenShift Container Platform バージョンを指します。

プラットフォーム	ホスティング OpenShift Container Platform のバージョン	ホステッド OpenShift Container Platform バージョン
ベアメタル	4.14 - 4.15	4.14 - 4.15 のみ
Red Hat OpenShift Virtualization	4.14 - 4.15	4.14 - 4.15 のみ
AWS (テクノロジープレビュー)	4.11 - 4.15	4.14 - 4.15 のみ

注記: Hosted control plane の同じプラットフォームで、ハブクラスターとワーカーを実行します。

コントロールプレーンは、単一の namespace に含まれる Pod として実行され、Hosted control plane クラスターに関連付けられます。OpenShift Container Platform がこのタイプのホステッドクラスターを作成すると、コントロールプレーンから独立したワーカーノードが作成されます。

Hosted control plane クラスターには、いくつかの利点があります。

- 専用コントロールプレーンノードをホストする必要がなくなるため、コストを節約できます。
- コントロールプレーンとワークロードを分離することで、分離が改善され、変更が必要になる設定エラーが減少します。

- コントロールプレーンノードのブートストラップの要件をなくすことで、クラスターの作成時間を短縮します。
- ターンキーデプロイメントまたは完全にカスタマイズされた OpenShift Container Platform プロビジョニングをサポートします。

hosted control plane を使用するには、次のドキュメントから始めてください。

1. Hosted control plane のサイジングに関するガイダンス
2. Hosted control plane コマンドラインインターフェイスのインストール
3. ホステッドクラスターのワークロードの分散

次に、使用する予定のプラットフォームに関連するドキュメントを参照してください。

- AWS での Hosted Control Plane クラスターの設定 (テクノロジープレビュー)
- ベアメタルでの Hosted control plane クラスターの設定
- 64 ビット x86 OpenShift Container Platform クラスターでのホスティングクラスターの設定による、IBM Power コンピュートノードの hosted control plane の作成 (テクノロジープレビュー)
- IBM Z コンピュートノード用の **x86** ベアメタル上でのホスティングクラスターの設定 (テクノロジープレビュー)
- OpenShift Virtualization での Hosted control plane クラスターの管理
- 非接続環境での Hosted control plane の設定
- ホステッドコントロール機能の無効化

追加のネットワーク、Guaranteed CPU、およびノードプールの仮想マシンスケジューリングを設定するには、次のドキュメントを参照してください。

- 追加のネットワーク、Guaranteed CPU、およびノードプールの仮想マシンのスケジュールを設定する

hosted control plane の追加リソースについては、次の OpenShift Container Platform ドキュメントを参照してください。

- Hosted control plane のアーキテクチャー
- Hosted control plane の一般的な概念とペルソナの用語集
- ホステッドクラスターの監視ダッシュボードの作成
- Hosted control plane のバックアップ、復元、障害復旧

1.7.2. Hosted control plane のサイジングに関するガイダンス

ホステッドクラスターのワークロードやワーカーノード数などの多くの要因が、一定数のコントロールプレーンノード内に収容できるホステッドクラスターの数に影響します。このサイジングガイドを使用して、ホステッドクラスターの容量計画に役立ちます。このガイダンスは、可用性の高い Hosted control plane トポロジーを前提としています。ロードベースのサイジングの例は、ベアメタルクラス

ターで測定されています。クラウドベースのインスタンスには、メモリーサイズなど、さまざまな制限要因が含まれる場合があります。高可用性の Hosted control plane トポロジーの詳細は、**ホステッドクラスターのワークロードの分散** を参照してください。

次のリソース使用率のサイズ測定をオーバーライドし、メトリクスサービスの監視を無効化できます。詳細は、**関連情報** セクションの **リソース使用率測定のオーバーライド** を参照してください。

OpenShift Container Platform バージョン 4.12.9 以降でテストされた、次の高可用性 Hosted control plane 要件を参照してください。

- 78 pods
- etcd 用の 3 つの 8 GiB PV
- 最小 vCPU: 約 5.5 コア
- 最小メモリー: 約 19 GiB

1.7.2.1. Pod の制限

各ノードの **maxPods** 設定は、コントロールプレーンノードに収容できるホストクラスターの数に影響します。すべてのコントロールプレーンノードの **maxPods** 値に注意することが重要です。高可用性の Hosted control plane ごとに約 75 個の Pod を計画します。

ベアメタルノードの場合、マシンに十分なリソースがある場合でも、Pod 要件を考慮すると、各ノードに約 3 つの Hosted control plane が使用されるため、デフォルトで **maxPods** 設定に 250 が指定されていることが制限要因となる可能性があります。**KubeletConfig** 値を設定して **maxPods** 値を 500 に設定すると、Hosted control plane の密度が増し、追加のコンピューティングリソースを活用できるようになります。詳細は、OpenShift Container Platform ドキュメントの **ノードあたりの最大 Pod 数の設定** を参照してください。

1.7.2.2. 要求ベースのリソース制限

クラスターがホストできる Hosted Control Plane の最大数は、Pod からの Hosted Control Plane CPU およびメモリー要求に基づいて計算されます。

可用性の高い Hosted Control Plane は、5 つの vCPU と 18 GB のメモリーを要求する 78 個の Pod で設定されています。これらのベースライン数値は、クラスターワーカーノードのリソース容量と比較され、Hosted Control Plane の最大数を推定します。

1.7.2.3. ロードベースの制限

クラスターがホストできる Hosted Control Plane の最大数は、ホストされたコントロールプレーンの Kubernetes API サーバーに何らかのワークロードが配置されている場合の Hosted Control Plane Pod の CPU とメモリーの使用率に基づいて計算されます。

ワークロードの増加に伴う Hosted Control Plane リソース使用率を測定するには、次の方法を使用します。

- KubeVirt プラットフォームを使用し、それぞれ 8 つの vCPU と 32 GiB を使用する 9 つのワーカーを持つホスト型クラスター
- 次の定義に基づいて、API コントロールプレーンのストレスに重点を置くように設定されたワークロードテストプロファイル:
    - 各 namespace にオブジェクトを作成し、合計 100 の namespace まで拡張しました。

- オブジェクトの継続的な削除と作成による API のストレスの増加
- ワークロードの 1 秒あたりのクエリー数 (QPS) とバースト設定を高く設定して、クライアント側のスロットリングを排除します。

負荷が 1000 QPS 増加すると、Hosted Control Plane リソースの使用率は 9 vCPU と 2.5 GB のメモリー増加します。

一般的なサイズ設定の目的で、**中程度の** ホストクラスター負荷である 1000 QPS API レートと、 **重い** ホストクラスター負荷である 2000 QPS API レートを検討してください。

注記: このテストは、予想される API 負荷に基づいてコンピューティングリソースの使用率を高めるための推定係数を提供します。正確な使用率は、クラスターのワークロードのタイプとペースによって異なる場合があります。

表1.7 ロードテーブル

Hosted control plane のリソース使用率のスケーリング	vCPU	メモリー (GiB)
負荷なしのリソース利用	2.9	11.1
1000 QPS でのリソース使用率	9.0	2.5

負荷が 1000 QPS 増加すると、Hosted Control Plane リソースの使用率は 9 vCPU と 2.5 GB のメモリー増加します。

一般的なサイジングの目的では、1000 QPS API レートを中程度のホステッドクラスターの負荷、2000 QPS API を高程度のホステッドクラスターの負荷とみなしてください。

次の例は、ワークロードおよび API レート定義の Hosted control plane リソースのスケーリングを示しています。

表1.8 API 料金表

QPS (API レート)	vCPU の使用量	メモリーの使用量 (GiB)
低負荷 (50 QPS 未満)	2.9	11.1
中負荷 (1000 QPS)	11.9	13.6
高負荷 (2000 QPS)	20.9	16.1

Hosted control plane のサイジングは、コントロールプレーンの負荷と、大量の API アクティビティー、etcd アクティビティー、またはその両方を引き起こすワークロードに関係します。データベースの実行など、データプレーンの負荷に重点を置くホスト型 Pod ワークロードでは、API レートが高い可能性があります。

1.7.2.4. サイジング計算の例

この例では、次のシナリオに対してサイジングのガイダンスを提供します。

- **hypershift.openshift.io/control-plane** ノードとしてラベル付けされたベアメタルワーカー 3 つ
- **maxPods** 値を 500 に設定している
- 負荷ベースの制限に応じて、予想される API レートは中または約 1000 である

表1.9 入力の制限

制限の説明	サーバー 1	サーバー 2
ワーカーノード上の vCPU 数	64	128
ワーカーノードのメモリー (GiB)	128	256
ワーカーあたりの最大 Pod 数	500	500
コントロールプレーンのホストに使用されるワーカーの数	3	3
最大 QPS ターゲットレート (1 秒あたりの API リクエスト)	1000	1000

表1.10 サイジング計算の例

ワーカーノードのサイズと API レートに基づいた計算値	サーバー 1	サーバー 2	計算の注記
vCPU リクエストに基づくワーカーあたりの最大ホストコントロールプレーン数	12.8	25.6	hosted control plane ごとにワーカー vCPU/5 合計 vCPU 要求の数
vCPU 使用率に基づくワーカーあたりの最大 Hosted control plane 数	5.4	10.7	vCPU の数 ÷ (2.9 測定されたアイドル状態の vCPU 使用率 + (QPS ターゲットレート ÷ 1000) × 9.0 1000 QPS 増加あたりの測定された vCPU 使用率)
メモリーリクエストに基づくワーカーごとの最大 Hosted control plane	7.1	14.2	hosted control plane ごとにワーカーメモリー GiB 0.2 18 GiB の合計メモリーリクエスト

メモリー使用量に基づくワーカーあたりの最大 Hosted control plane 数	9.4	18.8	ワーカーメモリー GiB ÷ (11.1 測定されたアイドルメモリー使用量 + (QPS ターゲットレート ÷ 1000) × 2.5 1000 QPS 増加あたりの測定されたメモリー使用量)
ノードごとの Pod の制限に基づくワーカーごとの最大 Hosted control plane	6.7	6.7	500 **maxPod**/Hosted control plane あたり 75 Pod
前述の最大値の中の最小値	5.4	6.7	
	vCPU の制限要因	**maxPods** の制限要因	
管理クラスター内の Hosted control plane の最大数	16	20	前述の最大 3 つのコントロールプレーンワーカーの最小数。

表1.11 Hosted control plane の容量メトリクス

名前	説明
mce_hs_addon_request_based_hcp_capacity_gauge	高可用性 Hosted Control Plane リソース要求に基づいて、クラスターがホストできる Hosted control plane の推定最大数。
mce_hs_addon_low_qps_based_hcp_capacity_gauge	すべての Hosted Control Plane がクラスターの Kube API サーバーに約 50 QPS を送信する場合、クラスターがホストできる Hosted Control Plane の推定最大数。
mce_hs_addon_medium_qps_based_hcp_capacity_gauge	すべての Hosted Control Plane がクラスターの Kube API サーバーに約 1000 QPS を送信する場合、クラスターがホストできる Hosted Control Plane の推定最大数。
mce_hs_addon_high_qps_based_hcp_capacity_gauge	すべての Hosted Control Plane がクラスターの Kube API サーバーに約 2000 QPS を送信する場合、クラスターがホストできる Hosted Control Plane の推定最大数。

mce_hs_addon_average_qps_based_hcp_capacity_gauge	Hosted Control Plane の既存の平均 QPS に基づいて、クラスターがホストできる Hosted Control Plane の推定最大数。アクティブな Hosted Control Plane がない場合は、QPS が低くなることが予想されます。

1.7.2.5. 関連情報

- ホステッドクラスターのワークロードの分散
- ノードあたりの Pod の最大数の設定
- リソース使用率測定のオーバーライド

1.7.3. リソース使用率測定のオーバーライド

リソース使用率のベースライン測定セットは、クラスターごとに異なる場合があります。詳細は、**Hosted Control Plane のサイズガイダンス** を参照してください。

クラスターのワークロードの種類とペースに基づいて、リソース使用率の測定をオーバーライドできます。以下の手順を実行します。

1. 次のコマンドを実行して、**ConfigMap** リソースを作成します。

   ```
   oc create -f <your-config-map-file.yaml>
   ```

   <your-config-map-file.yaml> は **hcp-sizing-baseline** config map を含む YAML ファイルの名前に置き換えます。

2. **local-cluster** namespace に **hcp-sizing-baseline** Config Map を作成し、上書きする測定値を指定します。ConfigMap は、次の YAML ファイルのようになります。

   ```
   kind: ConfigMap
   apiVersion: v1
   metadata:
     name: hcp-sizing-baseline
     namespace: local-cluster
   data:
     incrementalCPUUsagePer1KQPS: "9.0"
     memoryRequestPerHCP: "18"
     minimumQPSPerHCP: "50.0"
   ```

3. 以下のコマンドを実行して **hypershift-addon-agent** デプロイメントを削除し、**hypershift-addon-agent** Pod を再起動します。

   ```
   oc delete deployment hypershift-addon-agent -n open-cluster-management-agent-addon
   ```

4. **hypershift-addon-agent** Pod ログを監視します。次のコマンドを実行して、オーバーライドされた測定値が config map 内で更新されていることを確認します。

   ```
   oc logs hypershift-addon-agent -n open-cluster-management-agent-addon
   ```

   ログは以下の出力のようになります。

```
2024-01-05T19:41:05.392Z INFO agent.agent-reconciler agent/agent.go:793 setting
cpuRequestPerHCP to 5
2024-01-05T19:41:05.392Z INFO agent.agent-reconciler agent/agent.go:802 setting
memoryRequestPerHCP to 18
2024-01-05T19:53:54.070Z INFO agent.agent-reconciler
agent/hcp_capacity_calculation.go:141 The worker nodes have 12.000000 vCPUs
2024-01-05T19:53:54.070Z INFO agent.agent-reconciler
agent/hcp_capacity_calculation.go:142 The worker nodes have 49.173369 GB memory
```

オーバーライドされた測定値が **hcp-sizing-baseline** config map で適切に更新されない場合、**hypershift-addon-agent** Pod ログに次のエラーメッセージが表示されることがあります。

```
2024-01-05T19:53:54.052Z ERROR agent.agent-reconciler agent/agent.go:788 failed to get
configmap from the hub. Setting the HCP sizing baseline with default values. {"error":
"configmaps \"hcp-sizing-baseline\" not found"}
```

1.7.3.1. メトリクスサービスモニタリングの無効化

hypershift-addon マネージドクラスターアドオンを有効にすると、メトリクスサービスモニタリングがデフォルトで設定され、OpenShift Container Platform モニタリングが **hypershift-addon** からメトリクスを収集できるようになります。次の手順を実行して、メトリクスサービスの監視を無効化できます。

1. 次のコマンドを実行して、ハブクラスターにログインします。

```
oc login
```

2. 次のコマンドを実行して、**hypershift-addon-deploy-config** アドオンデプロイメント設定仕様を開いて編集します。

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

3. 次の例に示すように、**disableMetrics=true** カスタマイズ変数を仕様に追加します。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  customizedVariables:
  - name: hcMaxNumber
    value: "80"
  - name: hcThresholdNumber
    value: "60"
  - name: disableMetrics
    value: "true"
```

4. 変更を保存します。カスタマイズ変数 **enableMetrics=true** は、新規および既存の **hypershift-addon** マネージドクラスターアドオンの両方のメトリクスサービス監視設定を無効にします。

1.7.3.2. 関連情報

- Hosted control plane のサイジングに関するガイダンス

1.7.4. Hosted control plane コマンドラインインターフェイスのインストール

次の手順を実行して、Hosted control plane コマンドラインインターフェイス (**hcp**) をインストールできます。

1. OpenShift Container Platform コンソールから、**Help icon** > **Command Line Tools** をクリックします。
2. お使いのプラットフォーム用の **Download hcp CLI** をクリックします。
3. 次のコマンドを実行して、ダウンロードしたアーカイブを解凍します。

```
tar xvzf hcp.tar.gz
```

4. 次のコマンドを実行して、バイナリーファイルを実行可能にします。

```
chmod +x hcp
```

5. 次のコマンドを実行して、バイナリーファイルをパス内のディレクトリーに移動します。

```
sudo mv hcp /usr/local/bin/.
```

hcp create cluster コマンドを使用して、ホステッドクラスターを作成および管理できるようになりました。使用可能なパラメーターをリストするには、次のコマンドを入力します。

```
hcp create cluster <platform> --help (1)
```

(1) サポートされているプラットフォームは、**aws**、**agent**、および **kubevirt** です。

1.7.4.1. CLI を使用した Hosted Control Plane コマンドラインインターフェイスのインストール

次の手順を実行すると、CLI を使用して Hosted Control Plane コマンドラインインターフェイス (**hcp**) をインストールできます。

1. 次のコマンドを実行して、**hcp** バイナリーをダウンロードするための URL を取得します。

```
oc get ConsoleCLIDownload hcp-cli-download -o json | jq -r ".spec"
```

2. 次のコマンドを実行して **hcp** バイナリーをダウンロードします。

```
wget <hcp_cli_download_url> (1)
```

(1) **hcp_cli_download_url** は、前の手順で取得した URL に置き換えます。

3. 次のコマンドを実行して、ダウンロードしたアーカイブを解凍します。

```
tar xvzf hcp.tar.gz
```

4. 次のコマンドを実行して、バイナリーファイルを実行可能にします。

```
chmod +x hcp
```

5. 次のコマンドを実行して、バイナリーファイルをパス内のディレクトリーに移動します。

```
sudo mv hcp /usr/local/bin/.
```

1.7.4.2. コンテンツゲートウェイを使用した Hosted Control Plane コマンドラインインターフェイスのインストール

コンテンツゲートウェイを使用して、Hosted Control Plane コマンドラインインターフェイス (**hcp**) をインストールできます。以下の手順を実行します。

1. コンテンツゲートウェイ に移動し、**hcp** バイナリーをダウンロードします。
2. 次のコマンドを実行して、ダウンロードしたアーカイブを解凍します。

```
tar xvzf hcp.tar.gz
```

3. 次のコマンドを実行して、バイナリーファイルを実行可能にします。

```
chmod +x hcp
```

4. 次のコマンドを実行して、バイナリーファイルをパス内のディレクトリーに移動します。

```
sudo mv hcp /usr/local/bin/.
```

hcp create cluster コマンドを使用して、ホステッドクラスターを作成および管理できるようになりました。使用可能なパラメーターをリストするには、次のコマンドを入力します。

```
hcp create cluster <platform> --help (1)
```

(1) サポートされているプラットフォームは、**aws**、**agent**、および **kubevirt** です。

1.7.5. ホステッドクラスターのワークロードの分散

OpenShift Container Platform の Hosted Control Plane を使い始める前に、ホスト型クラスター Pod をインフラストラクチャーノードにスケジュールするためにノードにラベルを付ける必要があります。

ノードのラベル付けにより、次の機能が保証されます。

- 高可用性と適切なワークロードのデプロイメント。たとえば、**node-role.kubernetes.io/infra** ラベルを設定して、OpenShift Container Platform サブスクリプションに control-plane ワークロード数が割り当てられないようにできます。
- コントロールプレーンのワークロードを管理クラスター内の他のワークロードから分離します。

重要: ワークロードには管理クラスターを使用しないでください。ワークロードは、コントロールプレーンが実行されるノード上で実行してはなりません。

1.7.5.1. 管理クラスターノードのラベルとテイント

管理クラスター管理者は、管理クラスターノードで次のラベルとテイントを使用して、コントロールプレーンのワークロードをスケジュールします。

- **hypershift.openshift.io/control-plane: true**: このラベルとテイントを使用して、Hosted control plane ワークロードの実行専用にノードを割り当てます。**true** を設定すると、コントロールプレーンノードを他のコンポーネントと共有することがなくなります。

- **hypershift.openshift.io/cluster: <hosted-control-plane-namespace>**: ノードを単一のホストされたクラスター専用にする場合は、このラベルとテイントを使用します。

コントロールプレーン Pod をホストするノードに以下のラベルを適用します。

- **node-role.kubernetes.io/infra**: このラベルを使用して、サブスクリプションにコントロールプレーンワークロード数が割り当てられないようにします。

- **topology.kubernetes.io/zone**: このラベルを管理クラスターノードで使用して、障害ドメイン全体に高可用性クラスターをデプロイします。ゾーンは、ゾーンが設定されているノードの場所、ラック名、またはホスト名である場合があります。

各ラックを管理クラスターノードのアベイラビリティーゾーンとして使用するには、次のコマンドを入力します。

+

```
oc label node/<management_node1_name> node/<management_node2_name>
topology.kubernetes.io/zone=<rack_name>
```

ホステッドクラスターの Pod には許容範囲があり、スケジューラーはアフィニティールールを使用して Pod をスケジュールします。スケジューラーは、**hypershift.openshift.io/control-plane** および **hypershift.openshift.io/cluster: <hosted_control_plane_namespace>** でラベル付けされたノードへの Pod のスケジューリングを優先します。

ControllerAvailabilityPolicy オプションには、**HighlyAvailable** を使用します。これは、Hosted control plane のコマンドラインインターフェイス (**hcp**) がデプロイメントするデフォルト値です。このオプションを使用する場合は、**topology.kubernetes.io/zone を** トポロジーキーとして設定することで、さまざまな障害ドメインにわたるホステッドクラスター内のデプロイメントごとに Pod をスケジュールできます。高可用性ではないコントロールプレーンはサポートされていません。

1.7.5.2. ホステッドクラスターのノードのラベル付け

重要: Hosted Control Plane をデプロイする前に、ノードにラベルを追加する必要があります。

ホストされたクラスターで実行している Pod をインフラストラクチャーノードにスケジュールするには、**HostedCluster** カスタムリソース (CR) に **role.kubernetes.io/infra: ""** ラベルを追加します。以下の例を参照してください。

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

1.7.5.3. 優先クラス

4 つの組み込み優先クラスは、ホステッドクラスター Pod の優先順位とプリエンプションに影響を与えます。管理クラスター内に Pod は、次の上位から下位の順序で作成できます。

- **hypershift-operator**: HyperShift Operator Pod。
- **hypershift-etcd**: etcd 用の Pod。
- **hypershift-api-critical**: API 呼び出しとリソース許可が成功するために必要な Pod。この優先クラスには、**kube-apiserver** (集約された API サーバー) や Web フックなどの Pod が含まれます。
- **hypershift-control-plane** : API クリティカルではないものの、クラスターバージョンの Operator など、高い優先順位が必要なコントロールプレーン内の Pod。

1.7.5.4. 関連情報

Hosted control plane の詳細は、次のトピックを参照してください。

- ベアメタルでの Hosted control plane クラスターの設定
- OpenShift Virtualization での Hosted control plane クラスターの管理
- AWS での Hosted Control Plane クラスターの設定 (テクノロジープレビュー)

1.7.6. AWS での Hosted Control Plane クラスターの設定 (テクノロジープレビュー)

Hosted control plane を設定するには、ホスティングクラスターとホステッドクラスターが必要です。**hypershift-addon** マネージドクラスターアドオンを使用して既存のマネージドクラスターに HyperShift Operator をデプロイすることにより、そのクラスターをホスティングクラスターとして有効にして、ホステッドクラスターを作成し始めることができます。**hypershift-addon** マネージドクラスターアドオンは、multicluster engine Operator 2.5 および Red Hat Advanced Cluster Management 2.10 ハブクラスターの **local-cluster** マネージドクラスターに対してデフォルトで有効になっています。

ホストされたクラスター は、ホスティングクラスターでホストされる API エンドポイントとコントロールプレーンを含む OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。マルチクラスターエンジンの Operator コンソールまたは hosted control plane のコマンドラインインターフェイス **hcp** を使用して、ホステッドクラスターを作成できます。ホステッドクラスターは、管理対象クラスターとして自動的にインポートされます。この自動インポート機能を無効にする場合は、**multicluster engine operator へのホストクラスターの自動インポートの無効化** を参照してください。

重要:

- 各ホステッドクラスターに、クラスター全体で一意の名前が必要です。multicluster engine Operator によってホストクラスターを管理するには、ホストクラスター名を既存のマネージドクラスターと同じにすることはできません。
- ホステッドクラスター名として **clusters** を使用しないでください。
- Hosted control plane の同じプラットフォームで、ハブクラスターとワーカーを実行します。
- ホステッドクラスターは、マルチクラスターエンジンの operator 管理クラスターの namespace には作成できません。

1.7.6.1. 前提条件

ホスティングクラスターを設定するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform クラスターにインストールされた Kubernetes Operator 2.5 以降のマルチクラスターエンジン。multicluster engine Operator は、Red Hat Advanced Cluster Management をインストールすると自動的にインストールされます。multicluster engine Operator は、Red Hat Advanced Cluster Management を使用せずに OpenShift Container Platform OperatorHub から Operator としてインストールすることもできます。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は、multicluster engine Operator 2.5 以降で自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

  ```
  oc get managedclusters local-cluster
  ```

- AWS コマンドラインインターフェイス
- Hosted Control Plane コマンドラインインターフェイス

hosted control plane の関連資料については、次のドキュメントを参照してください。

- Hosted control plane 機能を無効にするか、すでに無効にしていて手動で有効にする場合は、Hosted control plane 機能の有効化または無効化 を参照してください。
- Red Hat Ansible Automation Platform ジョブを実行してホステッドクラスターを管理するには、ホステッドクラスターで実行するための Ansible Automation Platform ジョブの設定 を参照してください。
- SR-IOV Operator をデプロイするには、Hosted control plane への SR-IOV Operator のデプロイ を参照してください。

1.7.6.2. Amazon Web Services S3 バケットと S3 OIDC シークレットの作成

AWS でホステッドクラスターを作成および管理する予定の場合は、次の手順を実行します。

1. クラスターの OIDC 検出ドキュメントをホストするためのパブリックアクセスを持つ S3 バケットを作成します。
   - us-east-1 リージョンにバケットを作成するには、次のコードを入力します。

     ```
     aws s3api create-bucket --bucket <your-bucket-name>
     aws s3api delete-public-access-block --bucket <your-bucket-name>
     echo '{
        "Version": "2012-10-17",
        "Statement": [
           {
              "Effect": "Allow",
              "Principal": "*",
              "Action": "s3:GetObject",
              "Resource": "arn:aws:s3:::<your-bucket-name>/*"
           }
        ]
     }' | envsubst > policy.json
     aws s3api put-bucket-policy --bucket <your-bucket-name> --policy file://policy.json
     ```

- us-east-1 リージョン以外のリージョンにバケットを作成するには、次のコードを入力します。

```
aws s3api create-bucket --bucket <your-bucket-name> \
  --create-bucket-configuration LocationConstraint=<region> \
  --region <region>
aws s3api delete-public-access-block --bucket <your-bucket-name>
echo '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*"
        }
    ]
}' | envsubst > policy.json
aws s3api put-bucket-policy --bucket <your-bucket-name> --policy file://policy.json
```

2. HyperShift Operator 用に **hypershift-operator-oidc-provider-s3-credentials** という名前の OIDC S3 シークレットを作成します。

3. シークレットを **local-cluster** namespace に保存します。

4. 次の表を参照して、シークレットに次のフィールドが含まれていることを確認します。

フィールド名	説明
bucket	ホステッドクラスターの OIDC 検出ドキュメントをホストするためのパブリックアクセスを備えた S3 バケットが含まれています。
credentials	バケットにアクセスできる **default** プロファイルの認証情報を含むファイルへの参照。デフォルトでは、HyperShift は **default** プロファイルのみを使用して **バケット** を操作します。
region	S3 バケットのリージョンを指定します。

次の例は、サンプルの AWS シークレットテンプレートを示しています。

```
oc create secret generic hypershift-operator-oidc-provider-s3-credentials --from-file=credentials=<path>/.aws/credentials --from-literal=bucket=<s3-bucket-for-hypershift> --from-literal=region=<region> -n local-cluster
```

注記: シークレットのリカバリーバックアップは自動的に有効になりません。以下のコマンドを実行して、障害復旧用に **hypershift-operator-oidc-provider-s3-credentials** シークレットのバックアップを有効にするラベルを追加します。

```
oc label secret hypershift-operator-oidc-provider-s3-credentials -n local-cluster cluster.open-cluster-management.io/backup=true
```

1.7.6.3. ルーティング可能なパブリックゾーンの作成

ゲストクラスター内のアプリケーションにアクセスするには、パブリックゾーンがルーティング可能である必要があります。パブリックゾーンが存在する場合は、この手順を省略します。省力しない場合は、パブリックゾーンが既存の機能に影響を及ぼします。

次のコマンドを実行して、クラスター DNS レコードのパブリックゾーンを作成します。

```
aws route53 create-hosted-zone --name <your-basedomain> --caller-reference $(whoami)-$(date --rfc-3339=date)
```

your-basedomain をベースドメインに置き換えます (例: **www.example.com)**。

1.7.6.4. 外部 DNS の有効化

Hosted control plane ではコントロールプレーンとデータプレーンが分離されているため、次の 2 つの独立した領域で DNS を設定できます。

- 次のドメインなど、ホステッドクラスター内のワークロードの Ingress: ***.apps.service-consumer-domain.com**
- サービスプロバイダードメインを介した API または OAUTH エンドポイントなど、管理クラスター内のサービスエンドポイントの受信: ***.service-provider-domain.com**

hostedCluster.spec.dns の入力は、ホステッドクラスター内のワークロードの Ingress を決定します。**hostedCluster.spec.services.servicePublishingStrategy.route.hostname** の入力は、管理クラスター内のサービスエンドポイントの Ingress を決定します。

外部 DNS は、**LoadBalancer** または **Route** の公開タイプを指定し、その公開タイプのホスト名を提供するホステッドクラスター **Services** の名前レコードを作成します。**Private** または **PublicAndPrivate** エンドポイントアクセスタイプを持つホステッドクラスターの場合、**APIServer** サービスと **OAuth** サービスのみがホスト名をサポートします。**Private** ホストクラスターの場合、DNS レコードは VPC 内の Virtual Private Cloud (VPC) エンドポイントのプライベート IP に解決されます。

Hosted control plane は、次の 4 つのサービスを公開します。

- **APIServer**
- **OAuthServer**
- **Konnectivity**
- **Ignition**

これらの各サービスは、**HostedCluster** 仕様の **servicePublishingStrategy** を使用して公開されます。デフォルトでは、**servicePublishingStrategy** の **LoadBalancer** および **Route** タイプの場合、次の 2 つの方法のいずれかでサービスを公開します。

- **LoadBalancer** タイプの **Service** ステータスにあるロードバランサーのホスト名を使用する
- **Route** の **status.host** フィールド

ただし、Hosted control plane をマネージドサービスコンテキストにデプロイメントする場合、これらの方法では、基盤となる管理クラスターの Ingress サブドメインが公開され、管理クラスターのライフサイクルとディザスターリカバリーのオプションが制限される可能性があります。

DNS 間接化が **LoadBalancer** および **Route** 公開タイプに階層化されている場合、マネージドサービス

オペレーターは、サービスレベルドメインを使用してすべてのパブリックホステッドクラスターサービスを公開できます。このアーキテクチャーでは、DNS 名を新しい **LoadBalancer** または **Route** に再マッピングできますが、管理クラスターの Ingress ドメインは公開されません。Hosted control plane は、外部 DNS を使用して間接層を実現します。

管理クラスターの **hypershift** namespace に **hypershift** Operator と一緒に **external-dns** をデプロイできます。外部 DNS は、**external-dns.alpha.kubernetes.io**/hostname アノテーションを持つ **Services** または **Routes** を監視します。このアノテーションは、レコードなどの **Service**、または CNAME レコードなどの **Route** を指す DNS レコードを作成するために使用されます。

1.7.6.4.1. 前提条件

Hosted control plane の外部 DNS を設定する前に、次の前提条件を満たす必要があります。

- 指定できる外部パブリックドメイン
- AWS Route53 管理コンソールへのアクセス

1.7.6.4.2. Hosted control plane の外部 DNS の設定

Hosted control plane クラスターをサービスレベル DNS (外部 DNS) でプロビジョニングする予定の場合は、次の手順を実行します。

1. HyperShift Operator の AWS 認証情報シークレットを作成し、**local-cluster** namespace で **hypershift-operator-external-dns-credentials** という名前を付けます。
2. 次の表を参照して、シークレットに必須フィールドが含まれていることを確認してください。

フィールド名	説明	任意または必須
provider	サービスレベル DNS ゾーンを管理する DNS プロバイダー。	必須
domain-filter	サービスレベルドメイン。	必須
credentials	すべての外部 DNS タイプをサポートする認証情報ファイル。	AWS キーを使用する場合はオプション
aws-access-key-id	認証情報アクセスキー ID。	AWS DNS サービスを使用する場合はオプション
aws-secret-access-key	認証情報アクセスキーのシークレット。	AWS DNS サービスを使用する場合はオプション

次の例は、サンプルの **hypershift-operator-external-dns-credentials** シークレットテンプレートを示しています。

```
oc create secret generic hypershift-operator-external-dns-credentials --from-literal=provider=aws --from-literal=domain-filter=service.my.domain.com --from-file=credentials=<credentials-file> -n local-cluster
```

注記: シークレットのリカバリーバックアップは自動的に有効になりません。**hypershift-operator-external-dns-credentials** シークレットを災害復旧用にバックアップできるようにするラベルを追加するには、次のコマンドを入力します。

```
oc label secret hypershift-operator-external-dns-credentials -n local-cluster cluster.open-cluster-management.io/backup=""
```

1.7.6.4.3. パブリック DNS ホストゾーンの作成

AWS Route 53 管理コンソールで外部 DNS ドメインフィルターとして使用するパブリック DNS ホストゾーンを作成できます。

1. Route 53 管理コンソールで、**Create hosted zone** をクリックします。
2. **Hosted zone configuration** ページでドメイン名を入力し、タイプとして **Publish hosted zone** が選択されていることを確認し、**Create hosted zone** をクリックします。
3. ゾーンが作成されたら、**Records** タブの **Value/Route traffic to** 列の値をメモします。
4. メインドメインで、DNS 要求を委任ゾーンにリダイレクトするための NS レコードを作成します。**Value** フィールドに、前の手順でメモした値を入力します。
5. **Create records** をクリックします。
6. 新しいサブゾーンにテストエントリーを作成し、次の例のような **dig** コマンドでテストすることにより、DNS ホストゾーンが機能していることを確認します。

```
dig +short test.user-dest-public.aws.kerberos.com
192.168.1.1
```

7. **LoadBalancer** サービスと **Route** サービスのホスト名を設定するホストクラスターを作成するには、次のコマンドを入力します。ここで、**external-dns-domain** は作成したパブリックホストゾーンと一致します。

```
hcp create cluster aws --name=example --endpoint-access=PublicAndPrivate --external-dns-domain=service-provider-domain.com ...
```

この例は、ホステッドクラスターの結果として生じる **services** ブロックを示しています。

```
  platform:
    aws:
      endpointAccess: PublicAndPrivate
...
  services:
  - service: APIServer
    servicePublishingStrategy:
      route:
        hostname: api-example.service-provider-domain.com
      type: Route
  - service: OAuthServer
    servicePublishingStrategy:
      route:
        hostname: oauth-example.service-provider-domain.com
      type: Route
  - service: Konnectivity
```

```
    servicePublishingStrategy:
      type: Route
  - service: Ignition
    servicePublishingStrategy:
      type: Route
```

コントロールプレーンオペレーターは、**Services** と **Routes** を作成するときに、**external-dns.alpha.kubernetes.io/hostname** アノテーションを付けます。値は、そのタイプの **servicePublishingStrategy** の **hostname** フィールドです。コントロールプレーンオペレーターは、その名前をサービスエンドポイントに使用し、ホスト名が設定されている場合、external-dns などの DNS レコードを作成できるメカニズムが存在することを期待します。

サービスレベルの DNS 間接化を使用できるのはパブリックサービスのみです。プライベートサービスは **hypershift.local** プライベートゾーンを使用します。特定のエンドポイントアクセスタイプに対してプライベートなサービスに **hostname** を設定することは無効です。

次の表は、サービスとエンドポイントの組み合わせに対して **hostname** を設定することが有効な場合を示しています。

サービス	Public	PublicAndPrivate	Private
APIServer	可能	可能	N
OAuthServer	可能	可能	N
Konnectivity	可能	N	N
Ignition	可能	N	N

1.7.6.4.4. コマンドラインインターフェイスと外部 DNS を使用したクラスターのデプロイ

外部パブリックホストゾーンがすでに存在する場合は、**hypershift** operator と **external-dns** operator をデプロイする必要があります。次のコマンドを入力して、**external-dns** Operator が実行中であり、内部フラグがパブリックホストゾーンを指していることを確認します。

```
export KUBECONFIG=<path_to_management_cluster_kubeconfig>
export AWS_CREDS=~/.aws/credentials
export REGION=<region>

hypershift create cluster aws \
    --aws-creds ${AWS_CREDS} \
    --instance-type m6i.xlarge \
    --region ${REGION} \
    --auto-repair \
    --generate-ssh \
    --name <cluster_name> \
    --namespace clusters \
    --base-domain service-consumer-domain.com \ 1
    --node-pool-replicas 2 \
    --pull-secret ${HOME}/pull_secret.json \
    --release-image quay.io/openshift-release-dev/ocp-release:4.13.0-ec.3-x86_64 \
    --external-dns-domain=service-provider-domain.com \ 2
    --endpoint-access=PublicAndPrivate 3
```

■

1. パブリックホストゾーン **service-consumer-domain.com** を指します。これは通常、サービス利用者が所有する AWS アカウント内にあります。
2. パブリック外部 DNS ホストゾーン **service-provider-domain.com** を指します。これは通常、サービスプロバイダーが所有する AWS アカウント内にあります。
3. **PublicAndPrivate** として設定します。外部 DNS は、 **Public** または **PublicAndPrivate** 設定でのみ使用できます。

1.7.6.5. AWS PrivateLink の有効化

PrivateLink を使用して AWS プラットフォームで Hosted control plane クラスターをプロビジョニングする予定の場合は、次の手順を実行します。

1. HyperShift Operator の AWS 認証情報シークレットを作成し、**hypershift-operator-private-link-credentials** という名前を付けます。シークレットは、ホスティングクラスターとして使用されるマネージドクラスターの namespace であるマネージドクラスター namespace に存在する必要があります。**local-cluster** を使用した場合は、**local-cluster** namespace にシークレットを作成します
2. シークレットに必要なフィールドが含まれることを確認するには、以下の表を参照してください。

フィールド名	説明	任意または必須
region	Private Link で使用するリージョン	必須
aws-access-key-id	認証情報アクセスキー ID。	必須
aws-secret-access-key	認証情報アクセスキーのシークレット。	必須

次の例は、サンプルの **hypershift-operator-private-link-credentials** シークレットテンプレートを示しています。

```
oc create secret generic hypershift-operator-private-link-credentials --from-literal=aws-access-key-id=<aws-access-key-id> --from-literal=aws-secret-access-key=<aws-secret-access-key> --from-literal=region=<region> -n local-cluster
```

注記: シークレットのリカバリーバックアップは自動的に有効になりません。以下のコマンドを実行して、障害復旧用に **hypershift-operator-private-link-credentials** シークレットのバックアップを有効にするラベルを追加します。

```
oc label secret hypershift-operator-private-link-credentials -n local-cluster cluster.open-cluster-management.io/backup=""
```

1.7.6.6. ホステッドクラスターのディザスタリカバリー

Hosted control plane は、multicluster engine Operator ハブクラスター上で実行されます。データプレーンは、選択した別のプラットフォーム上で実行されます。マルチクラスターエンジンの operator

ハブクラスターを災害から復旧する場合、ホストされているコントロールプレーンも復旧する必要がある場合があります。

Hosted control plane クラスターをバックアップし、別のクラスターに復元する方法は、AWS リージョン内のホステッドクラスターのディザスターリカバリー を参照してください。

重要: ホステッドクラスターの障害復旧は AWS でのみ利用できます。

1.7.6.7. ホステッドクラスターの AWS へのデプロイ

Hosted control plane マンドラインインターフェイス (**hcp**) を設定し、**local-cluster** ホスティングクラスターとして有効にしたら、次の手順を実行して、AWS にホステッドクラスターをデプロイできます。プライベートホストクラスターをデプロイするには、**AWS でのプライベートホストクラスターのデプロイ** を参照してください。

1. 各変数の説明を表示するには、次のコマンドを実行します。

```
hcp create cluster aws --help
```

2. ハブクラスターにログインしていることを確認します。

3. 次のコマンドを実行して、ホステッドクラスターを作成します。

```
hcp create cluster aws \
  --name <hosted_cluster_name> \ 1
  --infra-id <infra_id> \ 2
  --base-domain <basedomain> \ 3
  --aws-creds <path_to_aws_creds> \ 4
  --pull-secret <path_to_pull_secret> \ 5
  --region <region> \ 6
  --generate-ssh \
  --node-pool-replicas <node_pool_replica_count> \ 7
  --namespace <hosted_cluster_namespace> 8
```

1. ホストされているクラスターの名前を指定します (例: **example**)。**<hosted_cluster_name>** と **<infra_id>** の値が同じであることを確認してください。そうしないと、クラスターが Kubernetes Operator コンソールのマルチクラスターエンジンに正しく表示されない可能性があります。
2. インフラストラクチャーの名前を指定します (例: **clc-name-hs1**)。
3. ベースドメインを指定します (例: **dev09.red-chesterfield.com**)。
4. AWS 認証情報ファイルへのパスを指定します (例: **/user/name/.aws/credentials**)。
5. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
6. AWS リージョン名を指定します (例: **us-east-1**)。
7. ノードプールのレプリカ数を指定します (例: **2**)。
8. ホストされたクラスターの namespace を指定します (例: **clusters**)。

   注記: デフォルトでは、**HostedCluster** と **NodePool** のすべてのカスタムリソースが **clusters** namespace に作成されます。**--namespace <namespace>** パラメーターを指定すると、選択した namespace に **HostedCluster** および **NodePool** カスタムリソースが作成されます。

1. 以下のコマンドを実行して、ホステッドクラスターのステータスを確認することもできます。

```
oc get hostedclusters -n <hosted_cluster_namespace>
```

2. 以下のコマンドを実行してノードプールを確認できます。

```
oc get nodepools --namespace <hosted_cluster_namespace>
```

1.7.6.8. AWS 上の複数のゾーンにホステッドクラスターを作成する

次のコマンドを入力して、パブリックゾーンのベースドメインを指定してクラスターを作成します。

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
--aws-creds <path-to-aws-credentials> \ 5
--region <region> \ 6
--zones <zones> 7
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ノードプールのレプリカ数を指定します (例: **2**)。
3. ベースドメインを指定します (例: **example.com**)。
4. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
5. AWS 認証情報ファイルへのパスを指定します (例: **/user/name/.aws/credentials**)。
6. AWS リージョン名を指定します (例: **us-east-1**)。
7. AWS リージョン内のアベイラビリティーゾーンを指定します (例: **us-east-1a**、**us-east-1b**)。

指定したゾーンごとに、次のインフラストラクチャーが作成されます。

- パブリックサブネット
- プライベートサブネット
- NAT ゲートウェイ
- プライベートルートテーブル (パブリックルートテーブルはパブリックサブネット間で共有されます)

ゾーンごとに 1 つの **NodePool** リソースが作成されます。ノードプール名の末尾にはゾーン名が付けられます。ゾーンのプライベートサブネットは **spec.platform.aws.subnet.id** に設定されます。

1.7.6.8.1. AWS でホストされたクラスターを作成するための認証情報の提供

hcp create cluster aws コマンドを使用してホステッドクラスターを作成する場合は、クラスターのインフラストラクチャーリソースを作成する権限が割り当てられた AWS アカウント認証情報を指定する必要があります。インフラストラクチャーリソースの例としては、VPC、サブネット、NAT ゲートウェイなどがあります。AWS 認証情報は、**--aws-creds** フラグを使用する方法と、Multi-Cluster Engine Operator からの AWS クラウドプロバイダーのシークレットを使用する方法の 2 つの方法で指定できます。

1.7.6.8.1.1. --aws-creds フラグを使用した認証情報の指定

--aws-creds フラグを使用して認証情報を指定する場合は、そのフラグを AWS 認証情報ファイルパスの値に使用します。

以下の例を参照してください。

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
--aws-creds <path-to-aws-credentials> \ 5
--region <region> 6
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ノードプールのレプリカ数を指定します (例: **2**)。
3. ベースドメインを指定します (例: **example.com**)。
4. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
5. AWS 認証情報ファイルへのパスを指定します (例: **/user/name/.aws/credentials**)。
6. AWS リージョン名を指定します (例: **us-east-1**)。

1.7.6.8.1.2. AWS クラウドプロバイダーシークレットを使用した認証情報の提供

シークレットには、SSH キー、プルシークレット、ベースドメイン、AWS 認証情報が含まれます。したがって、**--secret-creds** フラグを指定した **hcp create cluster aws** コマンドを使用して、AWS 認証情報を提供できます。以下の例を参照してください。

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--region <region> \ 2
--namespace <hosted-cluster-namespace> \ 3
--secret-creds <my-aws-cred> 4
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. AWS リージョン名を指定します (例: **us-east-1**)。
3. シークレットがデフォルトの **clusters** 名前空間にない場合は、ホストされているクラスター名前空間を指定します。

4 AWS シークレット名を指定します (例: **my-aws-cred**)。

このシークレットを使用すると、以下のフラグはオプションです。これらのフラグを **--secret-creds** フラグと合わせて指定すると、クラウドプロバイダーのシークレットの値よりも優先されます。

- **--aws-creds**
- **--base-domain**
- **--pull-secret**
- **--ssh-key**

1. {mce-shortF} コンソールを使用してシークレットを作成するには、ナビゲーションメニューから **Credentials** を選択し、コンソールで認証情報の作成手順に従います。
2. コマンドラインでシークレットを作成するには、次のコマンドを入力します。

```
$ oc create secret generic <my-secret> -n <namespace> --from-literal=baseDomain=<your-basedomain> --from-literal=aws_access_key_id=<your-aws-access-key> --from-literal=aws_secret_access_key=<your-aws-secret-key> --from-literal=pullSecret='{"auths": {"cloud.openshift.com":{"auth":"<auth>", "email":"<your-email>"}, "quay.io":{"auth":"<auth>", "email":"<your-email>"} } }' --from-literal=ssh-publickey=<your-ssh-publickey> --from-literal=ssh-privatekey=<your-ssh-privatekey>
```

シークレットの形式は次のとおりです。

```
apiVersion: v1
metadata:
  name: my-aws-cred 1
  namespace: clusters 2
type: Opaque
kind: Secret
stringData:
  ssh-publickey:         # Value
  ssh-privatekey:        # Value
  pullSecret:           # Value, required
  baseDomain:            # Value, required
  aws_secret_access_key:  # Value, required
  aws_access_key_id:      # Value, required
```

1.7.6.8.2. 関連情報

ホステッドクラスターに AWS Elastic File Service (EFS) CSI Driver Operator をインストールする手順は、セキュリティートークンサービスを使用した AWS EFS CSI Driver Operator の設定 を参照してください。

1.7.6.9. ARM64 OpenShift Container Platform クラスターで Hosted control plane を有効にする (テクノロジープレビュー)

ARM64 でホストされるコントロールプレーンを有効にして、管理クラスター環境で OpenShift Container Platform ARM64 データプレーンと連携できるようにすることができます。この機能は、AWS 上の Hosted control plane でのみ利用できます。

1.7.6.9.1. 前提条件

64 ビット ARM インフラストラクチャーにインストールされた OpenShift Container Platform クラスターが必要です。詳細は、OpenShift クラスターの作成: AWS (ARM) を参照してください。

1.7.6.9.2. ARM64 OpenShift Container Platform クラスター上でホステッドクラスターを実行する

ARM64 OpenShift Container Platform クラスター上でホステッドクラスターを実行するには、次の手順を完了します。

1. デフォルトのリリースイメージをマルチアーキテクチャーリリースイメージでオーバーライドするホステッドクラスターを作成します。
   たとえば、Hosted control plane コマンドラインインターフェイス (**hcp**) を介して、次のコマンドを入力します。クラスター名、ノードプールレプリカ、ベースドメイン、プルシークレット、AWS 認証情報、およびリージョンを実際の情報に置き換えます。

   ```
   hcp create cluster aws \
   --name $CLUSTER_NAME \
   --node-pool-replicas=$NODEPOOL_REPLICAS \
   --base-domain $BASE_DOMAIN \
   --pull-secret $PULL_SECRET \
   --aws-creds $AWS_CREDS \
   --region $REGION \
   --release-image quay.io/openshift-release-dev/ocp-release:4.13.0-rc.0-multi
   ```

   この例では、**--node-pool-replicas** フラグを使用してデフォルトの **NodePool** オブジェクトを追加します。

2. 64 ビット x_86 **NodePool** オブジェクトをホステッドクラスターに追加します。
   たとえば、Hosted control plane (**hcp**) コマンドラインインターフェイスを使用して、次のコマンドを入力します。クラスター名、ノードプール名、ノードプールレプリカを実際の情報に置き換えるよう注意してください。

   ```
   hcp create nodepool aws \
   --cluster-name $CLUSTER_NAME \
   --name $NODEPOOL_NAME \
   --node-count=$NODEPOOL_REPLICAS
   ```

1.7.6.9.3. AWS がホストするクラスターでの ARM NodePool オブジェクトの作成

同じ Hosted control plane から、64 ビット ARM および AMD64 上のアプリケーションワークロード (**NodePool** オブジェクト) をスケジュールできます。これを行うには、**NodePool** 仕様で **Arch** フィールドを定義し、**NodePool** オブジェクトに必要なプロセッサーアーキテクチャーを設定します。**arch** フィールドの有効な値は次のとおりです。

- **arm64**
- **amd64**

arch フィールドの値を指定しない場合は、デフォルトで **amd64** 値が使用されます。

AWS 上のホストされたクラスター上に ARM **NodePool** オブジェクトを作成するには、次の手順を実行します。

1. **HostedCluster** ホスト リソースで使用するマルチアーキテクチャーイメージがあることを確

l. **HostedCluster** カスタムリソースで使用するマルチアーキテクチャーイメージがあることを確認してください。マルチアーキテクチャーの夜間イメージには https://multi.ocp.releases.ci.openshift.org/ でアクセスできます。
マルチアーキテクチャーの夜間イメージは次の例のようになります。

```
% oc image info quay.io/openshift-release-dev/ocp-release-nightly@sha256:9b992c71f77501678c091e3dc77c7be066816562efe3d352be18128b8e8fce94 -a ~/pull-secrets.json

error: the image is a manifest list and contains multiple images - use --filter-by-os to select from:

  OS           DIGEST
  linux/amd64
sha256:c9dc4d07788ebc384a3d399a0e17f80b62a282b2513062a13ea702a811794a60
  linux/ppc64le
sha256:c59c99d6ff1fe7b85790e24166cfc448a3c2ac3ef3422fce3c7259e48d2c9aab
  linux/s390x
sha256:07fcd16d5bee95196479b1e6b5b3b8064dd5359dac75e3d81f0bd4be6b8fe208
  linux/arm64
sha256:1d93a6beccc83e2a4c56ecfc37e921fe73d8964247c1a3ec34c4d66f175d9b3d
```

2. Hosted control plane のコマンドラインインターフェイス (**hcp**) で次のコマンドを入力して、**NodePool** オブジェクトをレンダリングします。

```
hcp create nodepool aws --cluster-name $CLUSTER_NAME --name $ARM64_NODEPOOL_NAME --node-count=$NODEPOOL_REPLICAS --render > arm_nodepool_spec.yml
```

このコマンドは、次の例に示すように、**NodePool** オブジェクトの CPU アーキテクチャーを指定する YAML ファイルを作成します。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hypershift-arm-us-east-1a
  namespace: clusters
spec:
  arch: amd64
  clusterName: hypershift-arm
  management:
    autoRepair: false
    upgradeType: Replace
  nodeDrainTimeout: 0s
  platform:
    aws:
      instanceProfile: hypershift-arm-2m289-worker
      instanceType: m5.large
      rootVolume:
        size: 120
        type: gp3
      securityGroups:
      - id: sg-064ea63968d258493
      subnet:
```

```
        id: subnet-02c74cf1cf1e7413f
    type: AWS
  release:
    image: quay.io/openshift-release-dev/ocp-release-
nightly@sha256:390a33cebc940912a201a35ca03927ae5b058fbdae9626f7f4679786cab4fb1c

  replicas: 3
status:
  replicas: 0
```

3. 次のコマンドを入力して、YAML ファイルの **arch** 値と **instanceType** 値を変更します。このコマンドでは、ARM インスタンスタイプは **m6g.large** ですが、どの ARM インスタンスタイプでも機能します。

```
sed 's/arch: amd64/arch: arm64/g; s/instanceType: m5.large/instanceType: m6g.large/g'
arm_nodepool_spec.yml > temp.yml && mv temp.yml arm_nodepool_spec.yml
```

4. 次のコマンドを入力して、レンダリングされた YAML ファイルをホステッドクラスターに適用します。

```
oc apply -f arm_nodepool_spec.yml
```

1.7.6.10. ホステッドクラスターへのアクセス

ホステッドクラスターにアクセスするには、**kubeconfig** ファイルと **kubeadmin** 認証情報をリソースから直接取得するか、**hcp** コマンドラインインターフェイスを使用して **kubeconfig** ファイルを生成します。

- リソースから **kubeconfig** ファイルと認証情報を直接取得し、ホステッドクラスターにアクセスするには、Hosted control plane クラスターのアクセスシークレットを理解しておく必要があります。シークレットは、ホステッドクラスター (ホスティング) namespace に保存されます。**ホステッドクラスター (ホスティング)** namespace にはホステッドクラスターリソースが含まれており、 **Hosted control plane** namespace では Hosted control plane が実行されます。シークレット名の形式は次のとおりです。
  - **kubeconfig** シークレット: **<hosted-cluster-namespace>-<name>-admin-kubeconfig** (clusters-hypershift-demo-admin-kubeconfig)
  - **kubeadmin** パスワードシークレット: **<hosted-cluster-namespace>-<name>-kubeadmin-password** (clusters-hypershift-demo-kubeadmin-password)
    kubeconfig シークレットには Base64 でエンコードされた **kubeconfig** フィールドが含まれており、これをデコードしてファイルに保存し、次のコマンドで使用できます。

    ```
    oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
    ```

  kubeadmin パスワードシークレットも Base64 でエンコードされます。これをデコードし、そのパスワードを使用して、ホステッドクラスターの API サーバーまたはコンソールにログインできます。

- **hcp** CLI を使用してホステッドクラスターにアクセスして **kubeconfig** ファイルを生成するには、次の手順を実行します。
  1. 次のコマンドを入力して、**kubeconfig** ファイルを生成します。

```
hcp create kubeconfig --namespace <hosted-cluster-namespace> --name <hosted-cluster-name> > <hosted-cluster-name>.kubeconfig
```

2. **kubeconfig** ファイルを保存した後、次のコマンド例を入力して、ホステッドクラスターにアクセスできます。

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
```

1.7.6.10.1. 関連情報

ホステッドクラスターへのアクセス後に、ノードプールをスケーリングしたり、ホステッドクラスターのノード自動スケーリングを有効にしたりできます。詳細は、以下のトピックを参照してください。

- ノードプールのスケーリング
- ホステッドクラスターのノード自動スケーリングの有効化

ホステッドクラスターのノード調整を設定するには、次のトピックを参照してください。

- ホステッドクラスターにおけるノードのチューニング設定
- カーネルブートパラメーターを設定することによる、ホステッドクラスターの高度なノードチューニング

1.7.6.11. プライベートのホステッドクラスターを AWS にデプロイする (テクノロジープレビュー)

Hosted control plane (**hcp**) コマンドラインインターフェイスを設定し、**ローカルクラスターを** ホスティングクラスターとして有効にすると、ホステッドクラスターまたはプライベートのホステッドクラスターを AWS にデプロイできます。パブリックのホステッドクラスターを AWS にデプロイするには、**AWS でのホステッドクラスターのデプロイ** を参照してください。

デフォルトでは、Hosted control plane のゲストクラスターは、パブリック DNS および管理クラスターのデフォルトルーターを通じてパブリックにアクセスできます。

AWS のプライベートクラスターの場合、ゲストクラスターとのすべての通信は AWS PrivateLink 経由で行われます。AWS でプライベートクラスターをサポートするように Hosted control plane を設定するには、次の手順を実行します。

重要: パブリッククラスターは任意のリージョンに作成できますが、プライベートクラスターは **--aws-private-region** で指定されたリージョンにのみ作成できます。

- 前提条件
- AWS 上でプライベートホストクラスターを作成する
- AWS 上のプライベートホスティングクラスターへのアクセス
- 関連情報

1.7.6.11.1. 前提条件

AWS のプライベートホストクラスターを有効にするには、まず AWS PrivateLink を有効にする必要があります。詳細は、AWS PrivateLink の有効化 を参照してください。

1.7.6.11.2. AWS 上でプライベートホストクラスターを作成する

1. 次のコマンドを入力して、プライベートクラスター IAM ポリシードキュメントを作成します。

```
cat << EOF >> policy.json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpcEndpointServiceConfiguration",
        "ec2:DescribeVpcEndpointServiceConfigurations",
        "ec2:DeleteVpcEndpointServiceConfigurations",
        "ec2:DescribeVpcEndpointServicePermissions",
        "ec2:ModifyVpcEndpointServicePermissions",
        "ec2:CreateTags",
        "elasticloadbalancing:DescribeLoadBalancers"
      ],
      "Resource": "\*"
    }
  ]
}
```

2. 次のコマンドを入力して、AWS で IAM ポリシーを作成します。

```
aws iam create-policy --policy-name=hypershift-operator-policy --policy-
document=file://policy.json
```

3. 次のコマンドを入力して、**hypershift-operator** IAM ユーザーを作成します。

```
aws iam create-user --user-name=hypershift-operator
```

4. 次のコマンドを入力して、ポリシーを **hypershift-operator** ユーザーにアタッチします。**<policy-arn>** は、作成したポリシーの ARN に置き換えます。

```
aws iam attach-user-policy --user-name=hypershift-operator --policy-arn=<policy-arn>
```

5. 次のコマンドを入力して、ユーザーの IAM アクセスキーを作成します。

```
aws iam create-access-key --user-name=hypershift-operator
```

6. 次のコマンドを入力して、プライベートホストクラスターを作成します。必要に応じて、変数を実際の値に置き換えます。

```
hcp create cluster aws \
--name <hosted-cluster-name> \ 1
--node-pool-replicas=<node-pool-replica-count> \ 2
--base-domain <basedomain> \ 3
--pull-secret <path-to-pull-secret> \ 4
--aws-creds <path-to-aws-credentials> \ 5
--region <region> \ 6
--endpoint-access Private 7
```

1 1 ホストされているクラスターの名前を指定します (例: **example**)。

2 2 ノードプールのレプリカ数を指定します (例: **3**)。

3 ベースドメインを指定します (例: **example.com**)。

4 プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。

5 AWS 認証情報ファイルへのパスを指定します (例: **/user/name/.aws/credentials**)。

6 AWS リージョン名を指定します (例: **us-east-1**)。

7 クラスターがパブリックかプライベートかを定義します。

クラスターの API エンドポイントには、プライベート DNS ゾーンを通じてアクセスできます。

- **api.<hosted-cluster-name>.hypershift.local**
- ***.apps.<hosted-cluster-name>.hypershift.local**

1.7.6.11.3. AWS 上のプライベートホスティングクラスターへのアクセス

踏み台インスタンスを使用してプライベートクラスターにアクセスできます。

1. 次のコマンドを入力して、踏み台インスタンスを起動します。

```
hypershift create bastion aws --aws-creds=<aws-creds> --infra-id=<infra-id> --region=
<region> --ssh-key-file=<ssh-key>
```

<ssh-key> を、踏み台に接続するための SSH 公開鍵ファイルに置き換えます。SSH 公開鍵ファイルのデフォルトの場所は **~/.ssh/id_rsa.pub** です。**<aws-creds>** を AWS 認証情報ファイルへのパスに置き換えます (例: **/user/name/.aws/credentials**)。

注記: hypershift CLI はダウンロードできません。次のコマンドを使用して、**hypershift** namespace に存在する HyperShift Operator Pod を使用して CLI を抽出してください。**<hypershift-operator-pod-name>** は、HyperShift Operator Pod 名に置き換えてください。

```
oc project hypershift
oc rsync <hypershift-operator-pod-name>:/usr/bin/hypershift-no-cgo .
mv hypershift-no-cgo hypershift
```

1. 次のコマンドを入力して、クラスターノードプール内のノードのプライベート IP を検索します。

```
aws ec2 describe-instances --filter="Name=tag:kubernetes.io/cluster/<infra-
id>,Values=owned" | jq '.Reservations[] | .Instances[] | select(.PublicDnsName=="") |
.PrivateIpAddress'
```

2. 次のコマンドを入力して、ノードにコピーできるクラスターの **kubeconfig** ファイルを作成します。

```
hcp create kubeconfig > <cluster-kubeconfig>
```

3. 次のコマンドを入力して、**create bastion** コマンドから出力された IP を使用して踏み台を介していずれかのノードに SSH 接続します。

```
ssh -o ProxyCommand="ssh ec2-user@<bastion-ip> -W %h:%p" core@<node-ip>
```

4. SSH シェルから、次のコマンドを入力して、**kubeconfig** ファイルの内容をノード上のファイルにコピーします。

```
mv <path-to-kubeconfig-file> <new-file-name>
```

5. 次のコマンドを入力して、kubeconfig ファイルをエクスポートします。

```
export KUBECONFIG=<path-to-kubeconfig-file>
```

6. 次のコマンドを入力して、ゲストクラスターのステータスを確認します。

```
oc get clusteroperators clusterversion
```

1.7.6.11.4. 関連情報

AWS でのパブリックホステッドクラスターのデプロイの詳細は、AWS でのホステッドクラスターのデプロイ を参照してください。

1.7.6.12. AWS インフラストラクチャーと Hosted control plane の IAM 権限の管理 (テクノロジープレビュー)

AWS で Red Hat OpenShift Container Platform のホストされているコントロールプレーンを使用する場合、インフラストラクチャーの要件はセットアップに応じて異なります。

- 前提条件
- AWS インフラストラクチャーの要件
- ID とアクセス管理の権限
- AWS インフラストラクチャーと IAM リソースを個別に作成する

1.7.6.12.1. 前提条件

Hosted control plane クラスターを作成する前に、Hosted control plane を設定する必要があります。詳細は、AWS での Hosted Control Plane クラスターの設定 (テクノロジープレビュー) を参照してください。

1.7.6.12.2. AWS インフラストラクチャーの要件

AWS で Hosted control plane を使用する場合、インフラストラクチャー要件は次のカテゴリーに当てはまります。

- 任意の AWS アカウントの HyperShift Operator に必要なマネージド外のインフラストラクチャー
- ホステッドクラスターの AWS アカウントで事前に必要なマネージド外のインフラストラクチャー

- 管理 AWS アカウント内の Hosted control plane 管理インフラストラクチャー
- ホステッドクラスター内の Hosted control plane 管理インフラストラクチャー AWS アカウント
- ホステッドクラスターの Kubernetes 管理インフラストラクチャー AWS アカウント

Prerequired とは、Hosted control plane が適切に動作するために AWS インフラストラクチャーが必要であることを意味します。**Unmanaged** とは、Operator またはコントローラーがインフラストラクチャーを作成しないことを意味します。次のセクションには、AWS リソースの作成に関する詳細が含まれています。

1.7.6.12.2.1. 任意の AWS アカウントの HyperShift Operator に必要なマネージド外のインフラストラクチャー

任意の AWS アカウントは、ホストされるコントロールプレーンサービスのプロバイダーに依存します。

自己管理型の Hosted control plane では、クラスターサービスプロバイダーが AWS アカウントを制御します。**クラスターサービスプロバイダー** は、クラスターコントロールプレーンをホストする管理者であり、アップタイムを行います。管理対象の Hosted control plane では、AWS アカウントは Red Hat に属します。

HyperShift Operator の必須の非管理インフラストラクチャーでは、管理クラスター AWS アカウントに次のインフラストラクチャー要件が適用されます。

- 1 つの S3 バケット
  - OpenID Connect (OIDC)
- ルート 53 のホステッドゾーン
  - ホステッドクラスターのプライベートおよびパブリックエントリーをホストするドメイン

1.7.6.12.2.2. ホステッドクラスターの AWS アカウントで事前に必要なマネージド外のインフラストラクチャー

インフラストラクチャーが事前に必要であり、ホステッドクラスター AWS アカウントで管理されていない場合、すべてのアクセスモードのインフラストラクチャー要件は次のとおりです。

- 1 つの VPC
- 1 つの DHCP オプション
- 2 つのサブネット
  - 内部データプレーンサブネットであるプライベートサブネット
  - データプレーンからインターネットへのアクセスを可能にするパブリックサブネット
- 1 つのインターネットゲートウェイ
- 1 つの Elastic IP
- 1 つの NAT ゲートウェイ
- 1 つのセキュリティーグループ (ワーカーノード)

- 2 つのルートテーブル (1 つはプライベート、もう 1 つはパブリック)
- 2 つの Route 53 のホステッドゾーン
- 次のアイテムに対して十分な割り当てがあります:
  - パブリックホストクラスター用の 1 つの Ingress サービスロードバランサー
  - プライベートホストクラスター用の 1 つのプライベートリンクエンドポイント

注記: プライベートリンクネットワーキングが機能するには、ホステッドクラスター AWS アカウントのエンドポイントゾーンが、管理クラスター AWS アカウントのサービスエンドポイントによって解決されるインスタンスのゾーンと一致する必要があります。AWS では、ゾーン名は **us-east-2b** などのエイリアスであり、異なるアカウントの同じゾーンにマップされるとは限りません。そのため、プライベートリンクが機能するには、管理クラスターのリージョンのすべてのゾーンにサブネットまたはワーカーが必要です。

1.7.6.12.2.3. 管理 AWS アカウント内の Hosted control plane 管理インフラストラクチャー

インフラストラクチャーが管理 AWS アカウントの Hosted control plane によって管理されている場合、インフラストラクチャーの要件は、クラスターがパブリック、プライベート、またはその組み合わせであるかによって異なります。

パブリッククラスターを使用するアカウントの場合、インフラストラクチャー要件は次のとおりです。

- ネットワークロードバランサー: ロードバランサー Kube API サーバー
  - Kubernetes がセキュリティーグループを作成する
- Volumes
  - etcd の場合 (高可用性に応じて 1 つまたは 3 つ)
  - OVN-Kube の場合

プライベートクラスターを使用するアカウントの場合、インフラストラクチャー要件は次のとおりです。

- ネットワークロードバランサー: ロードバランサーのプライベートルーター
- エンドポイントサービス (プライベートリンク)

パブリッククラスターとプライベートクラスターを持つアカウントの場合、インフラストラクチャー要件は次のとおりです。

- ネットワークロードバランサー: ロードバランサーのパブリックルーター
- ネットワークロードバランサー: ロードバランサーのプライベートルーター
- エンドポイントサービス (プライベートリンク)
- Volumes:
  - etcd の場合 (高可用性に応じて 1 つまたは 3 つ)
  - OVN-Kube の場合

1.7.6.12.2.4. ホステッドクラスター内の Hosted control plane 管理インフラストラクチャー AWS アカウント

インフラストラクチャーがホステッドクラスター AWS アカウントの Hosted control plane によって管理されている場合、インフラストラクチャー要件は、クラスターがパブリック、プライベート、またはその組み合わせであるかによって異なります。

パブリッククラスターを使用するアカウントの場合、インフラストラクチャー要件は次のとおりです。

- ノードプールには、**Role** と **RolePolicy** が定義された EC2 インスタンスが必要です。

プライベートクラスターを使用するアカウントの場合、インフラストラクチャー要件は次のとおりです。

- アベイラビリティーゾーンごとに 1 つのプライベートリンクエンドポイント
- ノードプールの EC2 インスタンス

パブリッククラスターとプライベートクラスターを持つアカウントの場合、インフラストラクチャー要件は次のとおりです。

- アベイラビリティーゾーンごとに 1 つのプライベートリンクエンドポイント
- ノードプールの EC2 インスタンス

1.7.6.12.2.5. ホステッドクラスターの Kubernetes 管理インフラストラクチャー AWS アカウント

Kubernetes がホステッドクラスター AWS アカウントでインフラストラクチャーを管理する場合、インフラストラクチャー要件は次のとおりです。

- デフォルトの Ingress 用のネットワークロードバランサー
- レジストリー用の S3 バケット

1.7.6.12.3. ID とアクセス管理 (IAM) 権限

Hosted control plane のコンテキストでは、コンシューマーは Amazon リソースネーム (ARN) ロールを作成する責任があります。**コンシューマー** は、アクセス許可ファイルを生成する自動プロセスです。コンシューマーはコマンドラインインターフェイスまたは OpenShift Cluster Manager である可能性があります。Hosted control plane は、最小特権コンポーネントの原則を尊重する粒度を有効にしようとします。つまり、すべてのコンポーネントが独自のロールを使用して AWS オブジェクトを操作または作成し、ロールは製品が正常に機能するために必要なものに限定されます。

コマンドラインインターフェイスで ARN ロールを作成する方法の例は、AWS インフラストラクチャーと IAM リソースを個別に作成するを参照してください。

ホステッドクラスターは ARN ロールを入力として受け取り、コンシューマーは各コンポーネントの AWS 権限設定を作成します。その結果、コンポーネントは STS および事前設定された OIDC IDP を通じて認証できるようになります。

次のロールは、コントロールプレーン上で実行され、データプレーン上で動作する、Hosted control plane の一部のコンポーネントによって消費されます。

- **controlPlaneOperatorARN**
- **imageRegistryARN**

- **ingressARN**
- **kubeCloudControllerARN**
- **nodePoolManagementARN**
- **storageARN**
- **networkARN**

次の例は、ホステッドクラスターからの IAM ロールへの参照を示しています。

```
...
endpointAccess: Public
  region: us-east-2
  resourceTags:
  - key: kubernetes.io/cluster/example-cluster-bz4j5
    value: owned
rolesRef:
    controlPlaneOperatorARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-control-plane-
operator
    imageRegistryARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-openshift-image-
registry
    ingressARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-openshift-ingress
    kubeCloudControllerARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-cloud-controller
    networkARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-cloud-network-config-
controller
    nodePoolManagementARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-node-pool
    storageARN: arn:aws:iam::820196288204:role/example-cluster-bz4j5-aws-ebs-csi-driver-controller
type: AWS
...
```

Hosted control plane が使用するロールを次の例に示します。

- **ingressARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticloadbalancing:DescribeLoadBalancers",
                "tag:GetResources",
                "route53:ListHostedZones"
            ],
            "Resource": "\*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "route53:ChangeResourceRecordSets"
            ],
            "Resource": [
                "arn:aws:route53:::PUBLIC_ZONE_ID",
                "arn:aws:route53:::PRIVATE_ZONE_ID"
```

```
            ]
        }
    ]
}
```

- **imageRegistryARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:PutBucketTagging",
                "s3:GetBucketTagging",
                "s3:PutBucketPublicAccessBlock",
                "s3:GetBucketPublicAccessBlock",
                "s3:PutEncryptionConfiguration",
                "s3:GetEncryptionConfiguration",
                "s3:PutLifecycleConfiguration",
                "s3:GetLifecycleConfiguration",
                "s3:GetBucketLocation",
                "s3:ListBucket",
                "s3:GetObject",
                "s3:PutObject",
                "s3:DeleteObject",
                "s3:ListBucketMultipartUploads",
                "s3:AbortMultipartUpload",
                "s3:ListMultipartUploadParts"
            ],
            "Resource": "\*"
        }
    ]
}
```

- **storageARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:CreateSnapshot",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:DeleteSnapshot",
                "ec2:DeleteTags",
                "ec2:DeleteVolume",
                "ec2:DescribeInstances",
                "ec2:DescribeSnapshots",
                "ec2:DescribeTags",
```

```
                "ec2:DescribeVolumes",
                "ec2:DescribeVolumesModifications",
                "ec2:DetachVolume",
                "ec2:ModifyVolume"
            ],
            "Resource": "\*"
        }
    ]
}
```

- **networkARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:DescribeInstanceTypes",
                "ec2:UnassignPrivateIpAddresses",
                "ec2:AssignPrivateIpAddresses",
                "ec2:UnassignIpv6Addresses",
                "ec2:AssignIpv6Addresses",
                "ec2:DescribeSubnets",
                "ec2:DescribeNetworkInterfaces"
            ],
            "Resource": "\*"
        }
    ]
}
```

- **kubeCloudControllerARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:DescribeInstances",
                "ec2:DescribeImages",
                "ec2:DescribeRegions",
                "ec2:DescribeRouteTables",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVolumes",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:CreateVolume",
                "ec2:ModifyInstanceAttribute",
                "ec2:ModifyVolume",
                "ec2:AttachVolume",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateRoute",
```

```
                "ec2:DeleteRoute",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteVolume",
                "ec2:DetachVolume",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:DescribeVpcs",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:AttachLoadBalancerToSubnets",
                "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancer",
                "elasticloadbalancing:CreateLoadBalancerPolicy",
                "elasticloadbalancing:CreateLoadBalancerListeners",
                "elasticloadbalancing:ConfigureHealthCheck",
                "elasticloadbalancing:DeleteLoadBalancer",
                "elasticloadbalancing:DeleteLoadBalancerListeners",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DetachLoadBalancerFromSubnets",
                "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
                "elasticloadbalancing:ModifyLoadBalancerAttributes",
                "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
                "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
                "elasticloadbalancing:AddTags",
                "elasticloadbalancing:CreateListener",
                "elasticloadbalancing:CreateTargetGroup",
                "elasticloadbalancing:DeleteListener",
                "elasticloadbalancing:DeleteTargetGroup",
                "elasticloadbalancing:DescribeListeners",
                "elasticloadbalancing:DescribeLoadBalancerPolicies",
                "elasticloadbalancing:DescribeTargetGroups",
                "elasticloadbalancing:DescribeTargetHealth",
                "elasticloadbalancing:ModifyListener",
                "elasticloadbalancing:ModifyTargetGroup",
                "elasticloadbalancing:RegisterTargets",
                "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
                "iam:CreateServiceLinkedRole",
                "kms:DescribeKey"
            ],
            "Resource": [
                "\*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

- **nodePoolManagementARN**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "ec2:AllocateAddress",
                "ec2:AssociateRouteTable",
                "ec2:AttachInternetGateway",
```

```
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:CreateInternetGateway",
            "ec2:CreateNatGateway",
            "ec2:CreateRoute",
            "ec2:CreateRouteTable",
            "ec2:CreateSecurityGroup",
            "ec2:CreateSubnet",
            "ec2:CreateTags",
            "ec2:DeleteInternetGateway",
            "ec2:DeleteNatGateway",
            "ec2:DeleteRouteTable",
            "ec2:DeleteSecurityGroup",
            "ec2:DeleteSubnet",
            "ec2:DeleteTags",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeAddresses",
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeImages",
            "ec2:DescribeInstances",
            "ec2:DescribeInternetGateways",
            "ec2:DescribeNatGateways",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeNetworkInterfaceAttribute",
            "ec2:DescribeRouteTables",
            "ec2:DescribeSecurityGroups",
            "ec2:DescribeSubnets",
            "ec2:DescribeVpcs",
            "ec2:DescribeVpcAttribute",
            "ec2:DescribeVolumes",
            "ec2:DetachInternetGateway",
            "ec2:DisassociateRouteTable",
            "ec2:DisassociateAddress",
            "ec2:ModifyInstanceAttribute",
            "ec2:ModifyNetworkInterfaceAttribute",
            "ec2:ModifySubnetAttribute",
            "ec2:ReleaseAddress",
            "ec2:RevokeSecurityGroupIngress",
            "ec2:RunInstances",
            "ec2:TerminateInstances",
            "tag:GetResources",
            "ec2:CreateLaunchTemplate",
            "ec2:CreateLaunchTemplateVersion",
            "ec2:DescribeLaunchTemplates",
            "ec2:DescribeLaunchTemplateVersions",
            "ec2:DeleteLaunchTemplate",
            "ec2:DeleteLaunchTemplateVersions"
        ],
        "Resource": [
            "\*"
        ],
        "Effect": "Allow"
    },
    {
        "Condition": {
            "StringLike": {
                "iam:AWSServiceName": "elasticloadbalancing.amazonaws.com"
```

```
            }
        },
        "Action": [
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": [
            "arn:*:iam::*:role/aws-service-
role/elasticloadbalancing.amazonaws.com/AWSServiceRoleForElasticLoadBalancing"
        ],
        "Effect": "Allow"
    },
    {
        "Action": [
            "iam:PassRole"
        ],
        "Resource": [
            "arn:*:iam::*:role/*-worker-role"
        ],
        "Effect": "Allow"
    }
  ]
}
```

- **controlPlaneOperatorARN**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateVpcEndpoint",
        "ec2:DescribeVpcEndpoints",
        "ec2:ModifyVpcEndpoint",
        "ec2:DeleteVpcEndpoints",
        "ec2:CreateTags",
        "route53:ListHostedZones"
      ],
      "Resource": "\*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets",
        "route53:ListResourceRecordSets"
      ],
      "Resource": "arn:aws:route53:::%s"
    }
  ]
}
```

1.7.6.12.4. AWS インフラストラクチャーと IAM リソースを個別に作成する

デフォルトでは、**hcp create cluster aws** コマンドは、ホステッドクラスターを使用してクラウドインフラストラクチャーを作成し、それを適用します。クラウドインフラストラクチャー部分を個別に作成

して、**hcp create cluster aws** コマンドをクラスターの作成のみに使用したり、クラスターを適用する前に変更できるようにレンダリングしたりすることができます。

クラウドインフラストラクチャー部分を個別に作成するには、AWS インフラストラクチャーを作成し、AWS Identity and Access (IAM) リソースを作成し、クラスターを作成する必要があります。

1.7.6.12.4.1. AWS インフラストラクチャーの作成

AWS インフラストラクチャーを作成するには、次のコマンドを入力します。

```
hypershift create infra aws --name CLUSTER_NAME \ 1
  --aws-creds AWS_CREDENTIALS_FILE \ 2
  --base-domain BASEDOMAIN \ 3
  --infra-id INFRA_ID \ 4
  --region REGION \ 5
  --output-file OUTPUT_INFRA_FILE 6
```

1. **CLUSTER_NAME** を、作成しているホステッドクラスターの名前に置き換えます。この値は、クラスターの Route 53 プライベートのホステッドゾーンを作成するために使用されます。
2. **AWS_CREDENTIALS_FILE** を、VPC、サブネット、NAT ゲートウェイなどのクラスターのインフラストラクチャーリソースを作成する権限を持つ AWS 認証情報ファイルの名前に置き換えます。この値は、ワーカーが存在するゲストクラスターの AWS アカウントに対応する必要があります。
3. **BASEDOMAIN** を、ホステッドクラスター Ingress に使用する予定のベースドメインの名前に置き換えます。この値は、レコードを作成できる Route 53 パブリックゾーンに対応している必要があります。
4. **INFRA_ID** をタグを使用してインフラストラクチャーを識別する一意の名前に置き換えます。この値は、Kubernetes のクラウドコントローラーマネージャーとクラスター API マネージャーによってクラスターのインフラストラクチャーを識別するために使用されます。通常、この値はクラスターの名前 (**CLUSTER_NAME**) に接尾辞を追加したものです。
5. **REGION** をクラスターのインフラストラクチャーを作成するリージョンに置き換えます。
6. **OUTPUT_INFRA_FILE** をインフラストラクチャーの ID を JSON 形式で保存するファイルの名前に置き換えます。このファイルを **hcp create cluster aws** コマンドへの入力として使用し、**HostedCluster** リソースと **NodePool** リソースのフィールドに値を設定できます。

注記: hypershift CLI はダウンロードできません。次のコマンドを使用して、**hypershift** namespace に存在する HyperShift Operator Pod を使用して CLI を抽出してください。**<hypershift-operator-pod-name>** は、HyperShift Operator Pod 名に置き換えてください。

+

```
oc project hypershift
oc rsync <hypershift-operator-pod-name>:/usr/bin/hypershift-no-cgo .
mv hypershift-no-cgo hypershift
```

コマンドを入力すると、次のリソースが作成されます。

- 1 つの VPC

- 1 つの DHCP オプション
- 1 つのプライベートサブネット
- 1 つのパブリックサブネット
- 1 つのインターネットゲートウェイ
- 1 つの NAT ゲートウェイ
- ワーカーノード用の 1 つのセキュリティーグループ
- 2 つのルートテーブル:1 つはプライベート、もう 1 つはパブリック
- 2 つのプライベートホストゾーン: クラスター Ingress 用に 1 つ、PrivateLink 用に 1 つ (プライベートクラスターを作成する場合)

これらのリソースにはすべて、**kubernetes.io/cluster/INFRA_ID=owned** タグが含まれています。ここで、**INFRA_ID** はコマンドで指定した値です。

1.7.6.12.4.2. AWS IAM リソースの作成

AWS IAM リソースを作成するには、次のコマンドを入力します。

```
hypershift create iam aws --infra-id INFRA_ID \ 1
  --aws-creds AWS_CREDENTIALS_FILE \ 2
  --oidc-storage-provider-s3-bucket-name OIDC_BUCKET_NAME \ 3
  --oidc-storage-provider-s3-region OIDC_BUCKET_REGION \ 4
  --region REGION \ 5
  --public-zone-id PUBLIC_ZONE_ID \ 6
  --private-zone-id PRIVATE_ZONE_ID \ 7
  --local-zone-id LOCAL_ZONE_ID \ 8
  --output-file OUTPUT_IAM_FILE 9
```

1. **INFRA_ID** を **create infra aws** コマンドで指定したのと同じ ID に置き換えます。この値は、ホステッドクラスターに関連付けられている IAM リソースを識別します。
2. **AWS_CREDENTIALS_FILE** をロールなどの IAM リソースを作成する権限を持つ AWS 認証情報ファイルの名前に置き換えます。このファイルは、インフラストラクチャーを作成するために指定した認証情報ファイルと同じである必要はありませんが、同じ AWS アカウントに対応している必要があります。
3. **OIDC_BUCKET_NAME** を、OIDC ドキュメントを保存するバケットの名前に置き換えます。このバケットは、Hosted control plane をインストールするための前提条件として作成されました。バケットの名前は、このコマンドによって作成される OIDC プロバイダーの URL を構築するために使用されます。
4. **OIDC_BUCKET_REGION** を、OIDC バケットが存在するリージョンに置き換えます。
5. **REGION** をクラスターのインフラストラクチャーが配置されているリージョンに置き換えます。この値は、ホステッドクラスターに属するマシンのワーカーインスタンスプロファイルを作成するために使用されます。
6. **PUBLIC_ZONE_ID** をゲストクラスターのパブリックゾーンの ID に置き換えます。この値は、

7 **PRIVATE_ZONE_ID** をゲストクラスターのプライベートゾーンの ID に置き換えます。この値は、Ingress Operator のポリシーを作成するために使用されます。この値は **create infra aws** コマンド

8 **LOCAL_ZONE_ID** は、プライベートクラスターの作成時にゲストクラスターのローカルゾーンの ID に置き換えます。この値は、コントロールプレーンオペレーターのポリシーを作成するために使用され、PrivateLink エンドポイントのレコードを管理できるようになります。この値は **create infra aws** コマンドによって生成される **OUTPUT_INFRA_FILE** で確認できます。

9 **OUTPUT_IAM_FILE** を IAM リソースの ID を JSON 形式で保存するファイルの名前に置き換えます。その後、このファイルを **hcp create cluster aws** コマンドへの入力として使用して、**HostedCluster** リソースと **NodePool** リソースのフィールドに値を設定できます。

コマンドを入力すると、次のリソースが作成されます。

- 1 つの OIDC プロバイダー。STS 認証を有効にするために必要です。
- 7 つのロール。Kubernetes コントローラーマネージャー、クラスター API プロバイダー、レジストリーなど、プロバイダーと対話するコンポーネントごとに分かれています。
- 1 つのインスタンスプロファイル。クラスターのすべてのワーカーインスタンスに割り当てられるプロファイルです。

1.7.6.12.4.3. クラスターの作成

クラスターを作成するには、次のコマンドを入力します。

```
hcp create cluster aws \
  --infra-id INFRA_ID \ 1
  --name CLUSTER_NAME \ 2
  --aws-creds AWS_CREDENTIALS \ 3
  --pull-secret PULL_SECRET_FILE \ 4
  --generate-ssh \ 5
  --node-pool-replicas 3
```

1 **INFRA_ID** を **create infra aws** コマンドで指定したのと同じ ID に置き換えます。この値は、ホステッドクラスターに関連付けられている IAM リソースを識別します。

2 **CLUSTER_NAME** を **create infra aws** コマンドで指定したのと同じ名前に置き換えます。

3 **AWS_CREDENTIALS** を **create infra aws** コマンドで指定したのと同じ値に置き換えます。

4 **PULL_SECRET_FILE** を有効な OpenShift Container Platform プルシークレットを含むファイルの名前に置き換えます。

5 **--generate-ssh** フラグはオプションですが、ワーカーに SSH 接続する必要がある場合に含めるとよいでしょう。SSH キーが生成され、ホステッドクラスターと同じ名 namespace にシークレットとして保存されます。

コマンドに **--render** フラグを追加して、クラスターに適用する前にリソースを編集できるファイルに出力をリダイレクトすることもできます。

コマンドを実行すると、次のリソースがクラスターに適用されます。

- namespace

- プルシークレットの秘密
- **HostedCluster**
- **NodePool**
- コントロールプレーンコンポーネントの 3 つの AWS STS シークレット
- **--generate-ssh** フラグを指定した場合は、1 つの SSH キーシークレット。

1.7.6.13. AWS でのホステッドクラスターの破棄

ホステッドクラスターとそのマネージドクラスターリソースを破棄するには、次の手順を実行します。

1. 次のコマンドを実行して、multicluster engine Operator のマネージドクラスターリソースを削除します。

   ```
   oc delete managedcluster <managed_cluster_name>
   ```

   ここで、**<managed_cluster_name>** は管理対象クラスターの名前です。

2. 次のコマンドを実行して、ホステッドクラスターとそのバックエンドリソースを削除します。

   ```
   hcp destroy cluster aws --name <hosted_cluster_name> --infra-id <infra_id> --aws-creds
   <path_to_aws_creds> --base-domain <basedomain>
   ```

   必要に応じて名前を置き換えます。

1.7.7. ベアメタルでの Hosted control plane クラスターの設定

ホスティングクラスターとして機能するようにクラスターを設定することで、Hosted control plane をデプロイメントできます。ホスティングクラスターは、コントロールプレーンがホストされる OpenShift Container Platform クラスターです。ホスティングクラスターは **管理** クラスターとも呼ばれます。

注記: 管理クラスターは、**マネージド** クラスターとは異なります。マネージドクラスターは、ハブクラスターが管理するクラスターです。

Hosted control plane 機能がデフォルトで有効になりました。

multicluster engine Operator 2.5 は、管理対象のハブクラスターであるデフォルトの **local-cluster** と、ホスティングクラスターとしてのハブクラスターのみをサポートします。Red Hat Advanced Cluster Management 2.10 では、マネージドハブクラスター (**local-cluster**) をホスティングクラスターとして使用できます。

ホストされたクラスター は、ホスティングクラスターでホストされる API エンドポイントとコントロールプレーンを含む OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。マルチクラスターエンジンの Operator コンソールまたは hosted control plane のコマンドラインインターフェイス **hcp** を使用して、ホステッドクラスターを作成できます。ホステッドクラスターは、管理対象クラスターとして自動的にインポートされます。この自動インポート機能を無効にする場合は、**multicluster engine operator へのホストクラスターの自動インポートの無効化** を参照してください。

重要:

- Hosted control plane の同じプラットフォームで、ハブクラスターとワーカーを実行します。

- 各ホステッドクラスターに、クラスター全体で一意の名前が必要です。multicluster engine Operator によってホストクラスターを管理するには、ホストクラスター名を既存のマネージドクラスターと同じにすることはできません。
- ホステッドクラスター名として **clusters** を使用しないでください。
- ホステッドクラスターは、マルチクラスターエンジンの operator 管理クラスターの namespace には作成できません。
- エージェントプラットフォームを使用して、Hosted control plane をベアメタルでプロビジョニングできます。エージェントプラットフォームは、Central Infrastructure Management サービスを使用して、ホステッドクラスターにワーカーノードを追加します。central infrastructure management の概要は、central infrastructure management の有効化 を参照してください。
- すべてのベアメタルホストでは、central infrastructure management が提供する検出イメージ ISO を使用して手動でブートする必要があります。ホストは手動で起動することも、**Cluster-Baremetal-Operator** を使用して自動化することもできます。各ホストが起動すると、エージェントプロセスが実行され、ホストの詳細が検出され、インストールが完了します。**Agent** カスタムリソースは、各ホストを表します。
- エージェントプラットフォームでホステッドクラスターを作成すると、HyperShift は Hosted Control Plane (HCP) namespace に Agent Cluster API プロバイダーをインストールします。
- ノードプールによってレプリカをスケーリングすると、マシンが作成されます。クラスター API プロバイダーは、すべてのマシンに対して、ノードプール仕様で指定された要件を満たすエージェントを見つけてインストールします。エージェントのステータスと状態を確認することで、エージェントのインストールを監視できます。
- ノードプールをスケールダウンすると、エージェントは対応するクラスターからバインド解除されます。エージェントを再利用するには、Discovery Image を使用してエージェントを再起動する必要があります。
- Hosted control plane のストレージを設定する場合は、etcd の推奨プラクティスを考慮してください。レイテンシー要件を満たすには、各コントロールプレーンノードで実行されるすべての Hosted control plane の etcd インスタンス専用の高速ストレージデバイスを使用します。LVM ストレージを使用して、ホストされた etcd Pod のローカルストレージクラスを設定できます。詳細は、OpenShift Container Platform ドキュメントの **推奨される etcd プラクティス** および **論理ボリュームマネージャーストレージを使用した永続ストレージ** を参照してください。

1.7.7.1. 前提条件

ホスティングクラスターを設定するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform クラスターにインストールされた Kubernetes Operator 2.2 以降のマルチクラスターエンジンが必要です。multicluster engine Operator は、Red Hat Advanced Cluster Management をインストールすると自動的にインストールされます。OpenShift Container Platform OperatorHub から Operator として Red Hat Advanced Cluster Management を使用せずに、multicluster engine Operator をインストールすることもできます。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は、multicluster engine Operator 2.2 以降で自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

```
oc get managedclusters local-cluster
```

- 管理クラスター上のベアメタルホストに **topology.kubernetes.io/zone** ラベルを追加する必要があります。そうしないと、ホストされるすべてのコントロールプレーン Pod が単一ノードでスケジュールされ、単一障害点が発生します。
- Central Infrastructure Management を有効にする必要があります。詳細は、Central Infrastructure Management サービスの有効化 を参照してください。
- Hosted control plane コマンドラインインターフェイスをインストールする 必要があります。

1.7.7.2. ベアメタルのファイアウォールとポートの要件

ポートが管理クラスター、コントロールプレーン、ホストクラスター間で通信できるように、ファイアウォールとポートの要件を満たしていることを確認します。

- **kube-apiserver** サービスはデフォルトでポート 6443 で実行され、コントロールプレーンコンポーネント間の通信には ingress アクセスが必要です。
  - **NodePort** 公開ストラテジーを使用する場合は、**kube-apiserver** サービスに割り当てられたノードポートが公開されていることを確認してください。
  - MetalLB ロードバランシングを使用する場合は、ロードバランサーの IP アドレスに使用される IP 範囲への ingress アクセスを許可します。
- **NodePort** 公開ストラテジーを使用する場合は、**ignition-server** および **Oauth-server** 設定にファイアウォールルールを使用します。
- **konnectivity** エージェントは、ホステッドクラスター上で双方向通信を可能にするリバーストンネルを確立し、ポート 6443 でクラスター API サーバーアドレスへの egress アクセスを必要とします。この egress アクセスを使用すると、エージェントは **kube-apiserver** サービスにアクセスできます。
  - クラスター API サーバーのアドレスが内部 IP アドレスの場合は、ワークロードサブネットからポート 6443 の IP アドレスへのアクセスを許可します。
  - アドレスが外部 IP アドレスの場合は、ノードからその外部 IP アドレスにポート 6443 で送信できるように許可します。
- デフォルトのポート 6443 を変更する場合は、その変更を反映するようにルールを調整します。
- クラスター内で実行されるワークロードに必要なポートがすべて開いていることを確認してください。
- ファイアウォールルール、セキュリティーグループ、またはその他のアクセス制御を使用して、必要なソースだけにアクセスを制限します。必要な場合を除き、ポートを公開しないでください。
- 実稼働環境の場合は、ロードバランサーを使用して、単一の IP アドレスによるアクセスを簡素化します。

1.7.7.3. ベアメタルインフラストラクチャーの要件

エージェントプラットフォームはインフラストラクチャーを作成しませんが、インフラストラクチャーに関して次の要件があります。

- エージェント: **エージェント** は Discovery Image で起動され、OpenShift Container Platform ノードとしてプロビジョニングする準備ができているホストを表します。
- DNS: API および Ingress エンドポイントは、ルーティング可能である必要があります。

ベアメタル上の hosted control plane の関連資料については、次のドキュメントを参照してください。

- etcd および LVM ストレージの推奨事項の詳細は、推奨される etcd プラクティス および 論理ボリュームマネージャーストレージを使用した永続ストレージ を参照してください。
- 非接続環境でベアメタル上に Hosted control plane を設定するには、非接続環境での Hosted control plane の設定 を参照してください。
- Hosted control plane 機能を無効にするか、すでに無効にしていて手動で有効にする場合は、Hosted control plane 機能の有効化または無効化 を参照してください。
- Red Hat Ansible Automation Platform ジョブを実行してホステッドクラスターを管理するには、ホステッドクラスターで実行するための Ansible Automation Platform ジョブの設定 を参照してください。
- SR-IOV Operator をデプロイするには、Hosted control plane への SR-IOV Operator のデプロイ を参照してください。
- この自動インポート機能を無効にする場合は、multicluster engine operator へのホストクラスターの自動インポートの無効化 を参照してください。

1.7.7.4. ベアメタルでの DNS の設定

ホステッドクラスターの API サーバーは、**NodePort** サービスとして公開されます。API サーバーに到達できる宛先を指す **api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN}** に、DNS エントリーが存在する必要があります。

DNS エントリーは、Hosted control plane を実行しているマネージドクラスター内のノードの 1 つを指すレコードと同様、単純化できます。エントリーは、受信トラフィックを Ingress Pod にリダイレクトするためにデプロイされるロードバランサーを指すこともできます。

- 次の DNS 設定の例を参照してください。

```
api.example.krnl.es.    IN A 192.168.122.20
api.example.krnl.es.    IN A 192.168.122.21
api.example.krnl.es.    IN A 192.168.122.22
api-int.example.krnl.es.    IN A 192.168.122.20
api-int.example.krnl.es.    IN A 192.168.122.21
api-int.example.krnl.es.    IN A 192.168.122.22
`*`.apps.example.krnl.es. IN A 192.168.122.23
```

- IPv6 ネットワークで非接続環境の DNS を設定する場合は、次の DNS 設定の例を参照してください。

```
api.example.krnl.es.    IN A 2620:52:0:1306::5
api.example.krnl.es.    IN A 2620:52:0:1306::6
api.example.krnl.es.    IN A 2620:52:0:1306::7
api-int.example.krnl.es.    IN A 2620:52:0:1306::5
api-int.example.krnl.es.    IN A 2620:52:0:1306::6
api-int.example.krnl.es.    IN A 2620:52:0:1306::7
`*`.apps.example.krnl.es. IN A 2620:52:0:1306::10
```

- デュアルスタックネットワークの非接続環境で DNS を設定する場合は、IPv4 と IPv6 の両方の DNS エントリーを含めるようにしてください。次の DNS 設定の例を参照してください。

```
host-record=api-int.hub-dual.dns.base.domain.name,192.168.126.10
host-record=api.hub-dual.dns.base.domain.name,192.168.126.10
address=/apps.hub-dual.dns.base.domain.name/192.168.126.11
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,192.168.126.20
dhcp-host=aa:aa:aa:aa:10:02,ocp-master-1,192.168.126.21
dhcp-host=aa:aa:aa:aa:10:03,ocp-master-2,192.168.126.22
dhcp-host=aa:aa:aa:aa:10:06,ocp-installer,192.168.126.25
dhcp-host=aa:aa:aa:aa:10:07,ocp-bootstrap,192.168.126.26

host-record=api-int.hub-dual.dns.base.domain.name,2620:52:0:1306::2
host-record=api.hub-dual.dns.base.domain.name,2620:52:0:1306::2
address=/apps.hub-dual.dns.base.domain.name/2620:52:0:1306::3
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,[2620:52:0:1306::5]
dhcp-host=aa:aa:aa:aa:10:02,ocp-master-1,[2620:52:0:1306::6]
dhcp-host=aa:aa:aa:aa:10:03,ocp-master-2,[2620:52:0:1306::7]
dhcp-host=aa:aa:aa:aa:10:06,ocp-installer,[2620:52:0:1306::8]
dhcp-host=aa:aa:aa:aa:10:07,ocp-bootstrap,[2620:52:0:1306::9]
```

次に、ベアメタルに Hosted control plane の ホストインベントリーを作成 します。

1.7.7.5. ベアメタルでのホステッドクラスターの作成

ベアメタルでホステッドクラスターを作成するか、インポートできます。ホステッドクラスターをインポートする手順は、**ホステッドクラスターのインポート** を参照してください。

1. 次のコマンドを入力して、Hosted Control Plane 名前空間を作成します。

```
oc create ns <hosted_cluster_namespace>-<hosted_cluster_name>
```

<hosted_cluster_namespace> を、ホストされたクラスターの名前空間名 (例: **clusters**) に置き換えます。**<hosted_cluster_name>** をホストされたクラスター名に置き換えます。

2. クラスターにデフォルトのストレージクラスが設定されていることを確認します。そうしないと、保留中の PVC が表示される場合があります。以下のコマンドを実行します。

```
hcp create cluster agent \
  --name=<hosted_cluster_name> \ 1
  --pull-secret=<path_to_pull_secret> \ 2
  --agent-namespace=<hosted_control_plane_namespace> \ 3
  --base-domain=<basedomain> \ 4
  --api-server-address=api.<hosted_cluster_name>.<basedomain> \
  --etcd-storage-class=<etcd_storage_class> \ 5
  --ssh-key  <path_to_ssh_public_key> \ 6
  --namespace <hosted_cluster_namespace> \ 7
  --control-plane-availability-policy SingleReplica \
  --release-image=quay.io/openshift-release-dev/ocp-release:<ocp_release_image> 8
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. プルシークレットへのパスを指定します (例: /**user/name/pullsecret**)。

3 Hosted Control Plane 名前空間を指定します (例: **clusters-example**)。**oc get agent -n <hosted_control_plane_namespace>** コマンドを使用して、この名前空間でエージェン

4 ベースドメインを指定します (例: **krnl.es**)。

5 etcd ストレージクラス名を指定します (例: **lvm-storageclass**)。

6 SSH 公開鍵へのパスを指定します。デフォルトのファイルパスは ~/**.ssh/id_rsa.pub** です。

7 ホストされたクラスターの名前空間を指定します。

8 使用するサポートされている OpenShift Container Platform のバージョンを指定します (例: **4.14.0-x86_64**)。非接続環境を使用している場合は、**<ocp_release_image>** をダイジェストイメージに置き換えます。OpenShift Container Platform リリースイメージダイジェストを抽出するには、**OpenShift Container Platform リリースイメージダイジェストの抽出** を参照してください。

3. しばらくしてから、次のコマンドを入力して、Hosted control plane の Pod が稼働中であることを確認します。

```
oc -n <hosted_control_plane_namespace> get pods
```

以下の出力例を参照してください。

```
NAME                                   READY   STATUS    RESTARTS   AGE
capi-provider-7dcf5fc4c4-nr9sq         1/1     Running   0          4m32s
catalog-operator-6cd867cc7-phb2q       2/2     Running   0          2m50s
certified-operators-catalog-884c756c4-zdt64   1/1     Running   0          2m51s
cluster-api-f75d86f8c-56wfz            1/1     Running   0          4m32s
```

1.7.7.5.1. コンソールを使用してベアメタル上にホストされたクラスターを作成する

1. OpenShift Container Platform Web コンソールを開き、管理者の認証情報を入力してログインします。コンソールを開く手順については、OpenShift Container Platform ドキュメントの Web コンソールへのアクセス を参照してください。

2. コンソールヘッダーで、**All Clusters** が選択されていることを確認します。

3. **Infrastructure > Clusters** をクリックします。

4. **Create cluster Host inventory > Hosted control plane**をクリックします。
Create cluster ページが表示されます。

5. **Create cluster** ページでプロンプトに従い、クラスター、ノードプール、ネットワーク、および自動化に関する詳細を入力します。
注: クラスターに関する詳細を入力する際には、次のヒントが役立つ場合があります。

   - 事前定義された値を使用してコンソールのフィールドに自動的に値を入力する場合は、ホストインベントリーの認証情報を作成できます。詳細は、**オンプレミス環境の認証情報の作成** を参照してください。

   - **Cluster details** ページのプルシークレットは、OpenShift Container Platform リソースへのアクセスに使用する OpenShift Container Platform プルシークレットです。ホストインベントリー認証情報を選択した場合は、プルシークレットが自動的に入力されます。

- **Node pools** ページでは、namespace にノードプールのホストが含まれます。コンソールを使用してホストインベントリーを作成した場合、コンソールは専用の namespace を作成します。

- **Networking** ページで、API サーバー公開ストラテジーを選択します。ホステッドクラスターの API サーバーは、既存のロードバランサーを使用するか、**NodePort** タイプのサービスとして公開できます。API サーバーに到達できる宛先を指す **api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN}** 設定に DNS エントリーを含める必要があります。このエントリーとして、管理クラスター内のノードの 1 つを指すレコード、または受信トラフィックを Ingress Pod にリダイレクトするロードバランサーを指すレコードを指定できます。

6. エントリーを確認し、**Create** をクリックします。
Hosted cluster ビューが表示されます。

7. **Hosted cluster** ビューでホストされたクラスターのデプロイメントを監視します。

8. ホストされたクラスターに関する情報が表示されない場合は、**All Clusters** が選択されていることを確認し、クラスター名をクリックします。

9. コントロールプレーンコンポーネントの準備が整うまで待ちます。このプロセスには数分かかる場合があります。

10. ノードプールのステータスを表示するには、**NodePool** セクションまでスクロールします。ノードをインストールするプロセスには約 10 分かかります。**Nodes** をクリックして、ノードがホストされたクラスターに参加したかどうかを確認することもできます。

1.7.7.5.2. ミラーレジストリーを使用してベアメタル上にホストされたクラスターを作成する

ミラーレジストリーを使用して、**hcp create cluster** コマンドで **--image-content-sources** フラグを指定して、ベアメタル上にホステッドクラスターを作成できます。以下の手順を実行します。

1. YAML ファイルを作成して、イメージコンテンツソースポリシー (ICSP) を定義します。以下の例を参照してください。

```
- mirrors:
  - brew.registry.redhat.io
  source: registry.redhat.io
- mirrors:
  - brew.registry.redhat.io
  source: registry.stage.redhat.io
- mirrors:
  - brew.registry.redhat.io
  source: registry-proxy.engineering.redhat.com
```

2. ファイルを **icsp.yaml** として保存します。このファイルにはミラーレジストリーが含まれます。

3. ミラーレジストリーを使用してホステッドクラスターを作成するには、次のコマンドを実行します。

```
hcp create cluster agent \
  --name=<hosted_cluster_name> \ 1
  --pull-secret=<path_to_pull_secret> \ 2
  --agent-namespace=<hosted_control_plane_namespace> \ 3
```

```
--base-domain=<basedomain> \ 4
--api-server-address=api.<hosted_cluster_name>.<basedomain> \
--image-content-sources icsp.yaml  \ 5
--ssh-key  <path_to_ssh_key> \ 6
--namespace <hosted_cluster_namespace> \ 7
--release-image=quay.io/openshift-release-dev/ocp-release:<ocp_release_image> 8
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
3. Hosted Control Plane 名前空間を指定します (例: **clusters-example**)。**oc get agent -n <hosted-control-plane-namespace>** コマンドを使用して、この名前空間でエージェントが使用可能であることを確認します。
4. ベースドメインを指定します (例: **krnl.es**)。
5. ICSP およびミラーレジストリーを定義する **icsp.yaml** ファイルを指定します。
6. SSH 公開鍵へのパスを指定します。デフォルトのファイルパスは **~/.ssh/id_rsa.pub** です。
7. ホストされたクラスターの名前空間を指定します。
8. 使用するサポートされている OpenShift Container Platform のバージョンを指定します (例: **4.14.0-x86_64**)。非接続環境を使用している場合は、**<ocp_release_image>** をダイジェストイメージに置き換えます。OpenShift Container Platform リリースイメージダイジェストを抽出するには、**OpenShift Container Platform リリースイメージダイジェストの抽出** を参照してください。

1.7.7.5.3. 関連情報

- コンソールでホステッドクラスターを作成するときに再利用できる認証情報を作成するには、オンプレミス環境の認証情報の作成 を参照してください。
- ホステッドクラスターをインポートするには、Hosted control plane クラスターの手動インポート を参照してください。
- ホステッドクラスターにアクセスするには、ホステッドクラスターへのアクセス を参照してください。
- Discovery Image を使用してホストインベントリーにホストを追加するには、Discovery Image を使用したホストインベントリーへのホストの追加 を参照してください。
- OpenShift Container Platform リリースイメージダイジェストを抽出するには、OpenShift Container Platform リリースイメージダイジェストの抽出 を参照してください。

1.7.7.6. ホステッドクラスター作成の確認

デプロイメントプロセスが完了したら、ホステッドクラスターが正常に作成されたことを確認できます。ホステッドクラスターの作成から数分後に、次の手順に従います。

1. 次の extract コマンドを入力して、新しいホステッドクラスターの kubeconfig を取得します。

```
oc extract -n <hosted-control-plane-namespace> secret/admin-kubeconfig --to=- >
kubeconfig-<hosted-cluster-name>
```

2. kubeconfig を使用して、ホステッドクラスターのクラスター Operator を表示します。以下のコマンドを入力します。

```
oc get co --kubeconfig=kubeconfig-<hosted-cluster-name>
```

以下の出力例を参照してください。

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
console                                    4.10.26   True        False         False      2m38s
dns                                        4.10.26   True        False         False      2m52s
image-registry                             4.10.26   True        False         False      2m8s
ingress                                    4.10.26   True        False         False      22m
```

3. 次のコマンドを入力して、ホステッドクラスター上で実行中の Pod を表示することもできます。

```
oc get pods -A --kubeconfig=kubeconfig-<hosted-cluster-name>
```

以下の出力例を参照してください。

```
NAMESPACE                                  NAME                                                READY
STATUS             RESTARTS       AGE
kube-system                                konnectivity-agent-khlqv                            0/1
Running            0              3m52s
openshift-cluster-node-tuning-operator     tuned-dhw5p                                         1/1
Running            0              109s
openshift-cluster-storage-operator         cluster-storage-operator-5f784969f5-vwzgz
1/1     Running            1 (113s ago)   20m
openshift-cluster-storage-operator         csi-snapshot-controller-6b7687b7d9-7nrfw
1/1     Running            0              3m8s
openshift-console                          console-5cbf6c7969-6gk6z                            1/1
Running            0              119s
openshift-console                          downloads-7bcd756565-6wj5j                          1/1
Running            0              4m3s
openshift-dns-operator                     dns-operator-77d755cd8c-xjfbn                       2/2
Running            0              21m
openshift-dns                              dns-default-kfqnh                                   2/2
Running            0              113s
```

1.7.7.7. ホステッドクラスターの NodePool オブジェクトのスケーリング

ホストされたクラスターにノードを追加することで、**NodePool** オブジェクトをスケールアップできます。

1. **NodePool** オブジェクトを 2 つのノードにスケーリングします。

```
oc -n <hosted-cluster-namespace> scale nodepool <nodepool-name> --replicas 2
```

ClusterAPI Agent エージェントプロバイダーは、ホステッドクラスターに割り当てられる 2 つ

のエージェントをランダムに選択します。これらのエージェントはさまざまな状態を経て、最終的に OpenShift Container Platform ノードとしてホステッドクラスターに参加します。エージェントは次の順序で状態を通過します。

- **binding**
- **discovering**
- **insufficient**
- **installing**
- **installing-in-progress**
- **added-to-existing-cluster**

2. 以下のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get agent
```

以下の出力例を参照してください。

```
NAME                                   CLUSTER         APPROVED   ROLE          STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6   hypercluster1   true       auto-assign
d9198891-39f4-4930-a679-65fb142b108b                   true       auto-assign
da503cf1-a347-44f2-875c-4960ddb04091   hypercluster1   true       auto-assign
```

3. 以下のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get agent -o jsonpath='{range .items[*]}BMH:
{@.metadata.labels.agent-install\.openshift\.io/bmh} Agent: {@.metadata.name} State:
{@.status.debugInfo.state}{"\n"}{end}'
```

以下の出力例を参照してください。

```
BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: binding
BMH: ocp-worker-0 Agent: d9198891-39f4-4930-a679-65fb142b108b State: known-unbound
BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: insufficient
```

4. 次の extract コマンドを入力して、新しいホステッドクラスターの kubeconfig を取得します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig -
-to=- > kubeconfig-<hosted-cluster-name>
```

5. エージェントが **added-to-existing-cluster** 状態に達したら、次のコマンドを入力して、ホステッドクラスターの OpenShift Container Platform ノードが表示されることを確認します。

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get nodes
```

以下の出力例を参照してください。

```
NAME          STATUS   ROLES    AGE     VERSION
ocp-worker-1   Ready    worker   5m41s   v1.24.0+3882f8f
ocp-worker-2   Ready    worker   6m3s    v1.24.0+3882f8f
```

Cluster Operator は、ワークロードをノードに追加することによって調整を開始します。

6. 次のコマンドを入力して、**NodePool** オブジェクトをスケールアップしたときに 2 台のマシンが作成されたことを確認します。

```
oc -n <hosted-control-plane-namespace> get machines
```

以下の出力例を参照してください。

```
NAME                    CLUSTER          NODENAME       PROVIDERID
PHASE     AGE   VERSION
hypercluster1-c96b6f675-m5vch   hypercluster1-b2qhl   ocp-worker-1   agent://da503cf1-
a347-44f2-875c-4960ddb04091   Running   15m   4.13z
hypercluster1-c96b6f675-tl42p   hypercluster1-b2qhl   ocp-worker-2   agent://4dac1ab2-
7dd5-4894-a220-6a3473b67ee6   Running   15m   4.13z
```

clusterversion 調整プロセスは最終的に、Ingress および Console クラスター Operator のみが欠落しているポイントに到達します。

7. 以下のコマンドを入力します。

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get clusterversion,co
```

以下の出力例を参照してください。

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version             False       True          40m     Unable to apply
4.13z: the cluster operator console has not yet successfully rolled out

NAME                                                VERSION   AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                                4.12z     False       False
False      11m     RouteHealthAvailable: failed to GET route (https://console-openshift-
console.apps.hypercluster1.domain.com): Get "https://console-openshift-
console.apps.hypercluster1.domain.com": dial tcp 10.19.3.29:443: connect: connection
refused
clusteroperator.config.openshift.io/csi-snapshot-controller                4.12z     True        False
False      10m
clusteroperator.config.openshift.io/dns                                    4.12z     True        False
False      9m16s
```

1.7.7.7.1. ノードプールの追加

名前、レプリカの数、およびエージェントラベルセレクターなどの追加情報を指定して、ホステッドクラスターのノードプールを作成できます。

1. ノードプールを作成するには、次の情報を入力します。

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"

hcp create nodepool agent \
  --cluster-name $CLUSTER_NAME \
```

```
--name $NODEPOOL_NAME \
--node-count $WORKER_COUNT \
--agentLabelSelector '{"matchLabels": {"size": "medium"}}' 1
```

1 **--agentLabelSelector** は任意です。ノードプールは、**"size" : "medium"** ラベルを持つエージェントを使用します。

2. **clusters** namespace 内のノードプールリソースをリストして、**nodepool** プールのステータスを確認します。

```
oc get nodepools --namespace clusters
```

3. 次のコマンドを入力して、**admin-kubeconfig** シークレットを抽出します。

```
oc extract -n <hosted-control-plane-namespace> secret/admin-kubeconfig --
to=./hostedcluster-secrets --confirm
```

以下の出力例を参照してください。

```
hostedcluster-secrets/kubeconfig
```

4. しばらくしてから、次のコマンドを入力してノードプールのステータスを確認できます。

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

5. 次のコマンドを入力して、使用可能なノードプールの数が予想されるノードプールの数と一致することを確認します。

```
oc get nodepools --namespace clusters
```

1.7.7.7.2. 関連情報

- データプレーンをゼロにスケールダウンするには、データプレーンをゼロにスケールダウンする を参照してください。

1.7.7.8. ベアメタル上のホステッドクラスターでの Ingress の処理

すべての OpenShift Container Platform クラスターには、通常、外部 DNS レコードが関連付けられているデフォルトのアプリケーション Ingress コントローラーがあります。たとえば、ベースドメイン **krnl.es** を使用して **example** という名前のホステッドクラスターを作成する場合は、ワイルドカードドメイン ***.apps.example.krnl.es** がルーティング可能であると予想できます。

***.apps** ドメインにロードバランサーとワイルドカード DNS レコードを設定するには、ゲストクラスターで次のアクションを実行します。

1. MetalLB Operator の設定が含まれる YAML ファイルを作成して、MetalLB をデプロイします。

```
apiVersion: v1
kind: Namespace
metadata:
  name: metallb
```

```
  labels:
    openshift.io/cluster-monitoring: "true"
  annotations:
    workload.openshift.io/allowed: management
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: metallb-operator-operatorgroup
  namespace: metallb
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: metallb-operator
  namespace: metallb
spec:
  channel: "stable"
  name: metallb-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

2. ファイルを **metallb-operator-config.yaml** として保存します。
3. 以下のコマンドを入力して設定を適用します。

```
oc apply -f metallb-operator-config.yaml
```

4. Operator の実行後、MetalLB インスタンスを作成します。
   a. MetalLB インスタンスの設定を含む YAML ファイルを作成します。

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb
```

   b. ファイルを **metallb-instance-config.yaml** として保存します。
   c. 次のコマンドを入力して、MetalLB インスタンスを作成します。

```
oc apply -f metallb-instance-config.yaml
```

5. 2 つのリソースを作成して MetalLB Operator を設定します。
   - 単一の IP アドレスを持つ **IPAddressPool** リソース。この IP アドレスは、クラスターノードが使用するネットワークと同じサブネット上にある必要があります。
   - **IPAddressPool** リソースが BGP プロトコルを通じて提供するロードバランサーの IP アドレスをアドバタイズするための **BGP** アドバタイズリソース。
     a. 設定を含む YAML ファイルを作成します。

```
apiVersion: metallb.io/v1beta1
```

```
kind: IPAddressPool
metadata:
  name: <ip_address_pool_name> 1
  namespace: metallb
spec:
  protocol: layer2
  autoAssign: false
  addresses:
    - <ingress_ip>-<ingress_ip> 2
---
apiVersion: metallb.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: <bgp_advertisement_name> 3
  namespace: metallb
spec:
  ipAddressPools:
    - <ip_address_pool_name> 4
```

1 4 **IPAddressPool** リソース名を指定します。

2 環境の IP アドレスを指定します（例： **192.168.122.23** ）。

3 **BGP アドバタイズ** リソース名を指定します。

a. ファイルを **ipaddresspool-bgpadvertisement-config.yaml** として保存します。

b. 次のコマンドを入力してリソースを作成します。

```
oc apply -f ipaddresspool-bgpadvertisement-config.yaml
```

1. **LoadBalancer** タイプのサービスを作成した後、MetalLB はサービスの外部 IP アドレスを追加します。

c. **metallb-loadbalancer-service.yaml** という名前の YAML ファイルを作成して、ingress トラフィックを Ingress デプロイメントにルーティングする新しいロードバランサーサービスを設定します。

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    metallb.universe.tf/address-pool: ingress-public-ip
  name: metallb-ingress
  namespace: openshift-ingress
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 80
    - name: https
      protocol: TCP
      port: 443
      targetPort: 443
```

```
    selector:
      ingresscontroller.operator.openshift.io/deployment-ingresscontroller: default
    type: LoadBalancer
```

d. **metallb-loadbalancer-service.yaml** ファイルを保存します。

e. 以下のコマンドを入力して YAML 設定を適用します。

```
oc apply -f metallb-loadbalancer-service.yaml
```

f. 次のコマンドを入力して、OpenShift Container Platform コンソールにアクセスします。

```
curl -kI https://console-openshift-console.apps.example.krnl.es

HTTP/1.1 200 OK
```

g. **clusterversion** と **clusteroperator** の値をチェックして、すべてが実行されていることを確認します。以下のコマンドを入力します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion,co
```

以下の出力例を参照してください。

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
clusterversion.config.openshift.io/version   4.x.y     True        False         3m32s   Cluster version
is 4.x.y

NAME                                                 VERSION   AVAILABLE   PROGRESSING
DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                    4.x.y     True        False
False      3m50s
clusteroperator.config.openshift.io/ingress                    4.x.y     True        False
False      53m
```

+ **4.x.y** は、使用する OpenShift Container Platform バージョン(**4.14.0-x86_64** など)に置き換えます。

1.7.7.8.1. 関連情報

- MetalLB の詳細は、OpenShift Container Platform ドキュメントの MetalLB および MetalLB Operator について を参照してください。

1.7.7.9. ホステッドクラスターのノード自動スケーリングの有効化

ホステッドクラスターにさらに容量が必要で、予備のエージェントが利用可能な場合は、自動スケーリングを有効にして新しいワーカーノードをインストールできます。

1. 自動スケーリングを有効にするには、次のコマンドを入力します。

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p
'[{"op": "remove", "path": "/spec/replicas"},{"op":"add", "path": "/spec/autoScaling", "value": {
"max": 5, "min": 2 }}]'
```

注記: この例では、ノードの最小数は 2、最大数は 5 です。追加できるノードの最大数は、プラットフォームによって制限される場合があります。たとえば、エージェントプラットフォームを使用する場合、ノードの最大数は使用可能なエージェントの数によって制限されます。

1. 新しいノードを必要とするワークロードを作成します。

   a. 次の例を使用して、ワークロード設定を含む YAML ファイルを作成します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: reversewords
  name: reversewords
  namespace: default
spec:
  replicas: 40
  selector:
    matchLabels:
      app: reversewords
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: reversewords
    spec:
      containers:
      - image: quay.io/mavazque/reversewords:latest
        name: reversewords
        resources:
          requests:
            memory: 2Gi
status: {}
```

   b. ファイルを **workload-config.yaml** として保存します。

   c. 以下のコマンドを入力して、YAML を適用します。

```
oc apply -f workload-config.yaml
```

2. 次のコマンドを入力して、**admin-kubeconfig** シークレットを抽出します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=./hostedcluster-secrets --confirm
```

   以下の出力例を参照してください。

```
hostedcluster-secrets/kubeconfig
```

3. 次のコマンドを入力して、新しいノードが **Ready** ステータスであるかどうかを確認できます。

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

4. ノードを削除するには、次のコマンドを入力してワークロードを削除します。

```
oc --kubeconfig ./hostedcluster-secrets -n default delete deployment reversewords
```

5. 数分間待ちます。その間に、容量の追加が必要にならないようにします。エージェントプラットフォームでは、エージェントは廃止され、再利用できます。次のコマンドを入力すると、ノードが削除されたことを確認できます。

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

1.7.7.9.1. ホストされたクラスターのノードの自動スケーリングを無効にする

ノードの自動スケーリングを無効にするには、次のコマンドを入力します。

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p '[\
{"op":"remove", "path": "/spec/autoScaling"}, \{"op": "add", "path": "/spec/replicas", "value": <specify-value-to-scale-replicas>]'
```

このコマンドは、YAML ファイルから **"spec.autoScaling"** を削除し、**"spec.replicas"** を追加し、**"spec.replicas"** を指定の整数値に設定します。

1.7.7.10. ベアメタルでのホステッドクラスターの破棄

コンソールを使用して、ベアメタルホステッドクラスターを破棄できます。ベアメタル上のホステッドクラスターを破壊するには、次の手順を実行します。

1. コンソールで、**Infrastructure** > **Clusters** に移動します。
2. **Clusters** ページで、破棄するクラスターを選択します。
3. **Actions** メニューで **Destroy clusters** を選択し、クラスターを削除します。

1.7.7.10.1. コマンドラインを使用したベアメタル上でのホステッドクラスターの破棄

ホステッドクラスターを破棄するには、次の手順を実行します。

- 次のコマンドを実行して、ホステッドクラスターとそのバックエンドリソースを削除します。

```
hcp destroy cluster agent --name <hosted_cluster_name>
```

<hosted_cluster_name> をホストされたクラスターの名前に置き換えます。

1.7.8. 非ベアメタルエージェントマシンを使用した Hosted Control Plane クラスターの設定 (テクノロジープレビュー)

ホスティングクラスターとして機能するようにクラスターを設定することで、Hosted control plane をデプロイメントできます。ホスティングクラスターは、コントロールプレーンがホストされる OpenShift Container Platform クラスターです。ホスティングクラスターは **管理** クラスターとも呼ばれます。

注記: 管理クラスターは、**マネージド** クラスターとは異なります。マネージドクラスターは、ハブクラスターが管理するクラスターです。

Hosted control plane 機能がデフォルトで有効になりました。

multicluster engine Operator 2.5 は、管理対象のハブクラスターであるデフォルトの **local-cluster** と、ホスティングクラスターとしてのハブクラスターのみをサポートします。Red Hat Advanced Cluster Management 2.10 では、マネージドハブクラスター (**local-cluster**) をホスティングクラスターとして使用できます。

ホストされたクラスター は、ホスティングクラスターでホストされる API エンドポイントとコントロールプレーンを含む OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。マルチクラスターエンジンの Operator コンソールまたは hosted control plane のコマンドラインインターフェイス **hcp** を使用して、ホステッドクラスターを作成できます。ホステッドクラスターは、管理対象クラスターとして自動的にインポートされます。この自動インポート機能を無効にする場合は、**multicluster engine operator へのホストクラスターの自動インポートの無効化** を参照してください。

重要:

- 各ホステッドクラスターに、クラスター全体で一意の名前が必要です。multicluster engine Operator によってホストクラスターを管理するには、ホストクラスター名を既存のマネージドクラスターと同じにすることはできません。
- ホステッドクラスター名として **clusters** を使用しないでください。
- Hosted control plane の同じプラットフォームで、ハブクラスターとワーカーを実行します。
- ホステッドクラスターは、マルチクラスターエンジンの operator 管理クラスターの namespace には作成できません。
- エージェントプラットフォームを使用して、エージェントマシンをワーカーノードとしてホステッドクラスターに追加できます。エージェントマシンは、Discovery Image でブートされ、OpenShift Container Platform ノードとしてプロビジョニングされる準備ができているホストを表します。エージェントプラットフォームは、Central Infrastructure Management サービスの一部です。詳細は、Central Infrastructure Management サービスの有効化 を参照してください。
- ベアメタルではないすべてのホストは、Central Infrastructure Management が提供する Discovery Image ISO を使用して手動でブートする必要があります。
- エージェントプラットフォームでホステッドクラスターを作成すると、HyperShift は Hosted Control Plane (HCP) namespace に Agent Cluster API プロバイダーをインストールします。
- ノードプールをスケールアップすると、レプリカごとにマシンが作成されます。Cluster API プロバイダーは、マシンごとに、承認済みで、検証に合格し、現在使用されておらず、ノードプール仕様で指定されている要件を満たしているエージェントを検索してインストールします。エージェントのステータスと状態を確認することで、エージェントのインストールを監視できます。
- ノードプールをスケールダウンすると、エージェントは対応するクラスターからバインド解除されます。エージェントを再利用するには、Discovery Image を使用してエージェントを再起動する必要があります。
- Hosted control plane のストレージを設定する場合は、etcd の推奨プラクティスを考慮してください。レイテンシー要件を満たすには、各コントロールプレーンノードで実行されるすべての Hosted control plane の etcd インスタンス専用の高速ストレージデバイスを使用します。LVM ストレージを使用して、ホストされた etcd Pod のローカルストレージクラスを設定できます。詳細は、OpenShift Container Platform ドキュメントの **推奨される etcd プラクティス** および **論理ボリュームマネージャーストレージを使用した永続ストレージ** を参照してください。

1.7.8.1. 前提条件

ホスティングクラスターを設定するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform クラスターにインストールされた Kubernetes Operator 2.5 以降のマルチクラスターエンジン。multicluster engine Operator は、Red Hat Advanced Cluster Management をインストールすると自動的にインストールされます。OpenShift Container Platform OperatorHub から Operator として Red Hat Advanced Cluster Management を使用せずに、multicluster engine Operator をインストールすることもできます。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

```
oc get managedclusters local-cluster
```

- Central Infrastructure Management を有効にする必要があります。詳細は、Central Infrastructure Management サービスの有効化 を参照してください。
- Hosted control plane コマンドラインインターフェイスをインストールする 必要があります。

1.7.8.2. 非ベアメタルエージェントマシンのファイアウォールとポートの要件

ポートが管理クラスター、コントロールプレーン、ホストクラスター間で通信できるように、ファイアウォールとポートの要件を満たしていることを確認します。

- **kube-apiserver** サービスはデフォルトでポート 6443 で実行され、コントロールプレーンコンポーネント間の通信には ingress アクセスが必要です。
  - **NodePort** 公開ストラテジーを使用する場合は、**kube-apiserver** サービスに割り当てられたノードポートが公開されていることを確認してください。
  - MetalLB ロードバランシングを使用する場合は、ロードバランサーの IP アドレスに使用される IP 範囲への ingress アクセスを許可します。
- **NodePort** 公開ストラテジーを使用する場合は、**ignition-server** および **Oauth-server** 設定にファイアウォールルールを使用します。
- **konnectivity** エージェントは、ホステッドクラスター上で双方向通信を可能にするリバーストンネルを確立し、ポート 6443 でクラスター API サーバーアドレスへの egress アクセスを必要とします。この egress アクセスを使用すると、エージェントは **kube-apiserver** サービスにアクセスできます。
  - クラスター API サーバーのアドレスが内部 IP アドレスの場合は、ワークロードサブネットからポート 6443 の IP アドレスへのアクセスを許可します。
  - アドレスが外部 IP アドレスの場合は、ノードからその外部 IP アドレスにポート 6443 で送信できるように許可します。
- デフォルトのポート 6443 を変更する場合は、その変更を反映するようにルールを調整します。
- クラスター内で実行されるワークロードに必要なポートがすべて開いていることを確認してください。

- ファイアウォールルール、セキュリティーグループ、またはその他のアクセス制御を使用して、必要なソースだけにアクセスを制限します。必要な場合を除き、ポートを公開しないでください。
- 実稼働環境の場合は、ロードバランサーを使用して、単一の IP アドレスによるアクセスを簡素化します。

1.7.8.3. 非ベアメタルエージェントマシンのインフラストラクチャー要件

エージェントプラットフォームはインフラストラクチャーを作成しませんが、インフラストラクチャーに関して次の要件があります。

- エージェント: **エージェント** は Discovery Image で起動され、OpenShift Container Platform ノードとしてプロビジョニングする準備ができているホストを表します。
- DNS: API および Ingress エンドポイントは、ルーティング可能である必要があります。

1.7.8.4. 非ベアメタルエージェントマシンでの DNS の設定

ホステッドクラスターの API サーバーは、**NodePort** サービスとして公開されます。API サーバーに到達できる宛先を指す **api.<hosted-cluster-name>.<basedomain>** に、DNS エントリーが存在する必要があります。

DNS エントリーは、Hosted control plane を実行しているマネージドクラスター内のノードの 1 つを指すレコードと同様、単純化できます。エントリーは、受信トラフィックを Ingress Pod にリダイレクトするためにデプロイされるロードバランサーを指すこともできます。

- 次の DNS 設定の例を参照してください。

```
api-int.example.krnl.es.    IN A 192.168.122.22
`*`.apps.example.krnl.es. IN A 192.168.122.23
```

- IPv6 ネットワークで非接続環境の DNS を設定する場合は、次の DNS 設定の例を参照してください。

```
api-int.example.krnl.es.    IN A 2620:52:0:1306::7
`*`.apps.example.krnl.es. IN A 2620:52:0:1306::10
```

- デュアルスタックネットワークの非接続環境で DNS を設定する場合は、IPv4 と IPv6 の両方の DNS エントリーを含めるようにしてください。次の DNS 設定の例を参照してください。

```
host-record=api-int.hub-dual.dns.base.domain.name,2620:52:0:1306::2
address=/apps.hub-dual.dns.base.domain.name/2620:52:0:1306::3
dhcp-host=aa:aa:aa:aa:10:01,ocp-master-0,[2620:52:0:1306::5]
```

1.7.8.5. 非ベアメタルエージェントマシンでのホステッドクラスターの作成

ホステッドクラスターの作成やインポートが可能です。ホステッドクラスターをインポートする手順は、**ホステッドクラスターのインポート** を参照してください。

1. 次のコマンドを入力して、Hosted Control Plane 名前空間を作成します。

```
oc create ns <hosted-cluster-namespace>-<hosted-cluster-name>
```

<hosted-cluster-namespace> を、ホストされたクラスターの名前空間名 (例: **clusters**) に置き換えます。**<hosted-cluster-name>** をホストされたクラスター名に置き換えます。

2. クラスターにデフォルトのストレージクラスが設定されていることを確認します。設定されていない場合は、PVC が保留になる可能性があります。次のコマンドを入力し、変数例を実際の情報に置き換えます。

```
hcp create cluster agent \
  --name=<hosted-cluster-name> \ 1
  --pull-secret=<path-to-pull-secret> \ 2
  --agent-namespace=<hosted-control-plane-namespace> \ 3
  --base-domain=<basedomain> \ 4
  --api-server-address=api.<hosted-cluster-name>.<basedomain> \
  --etcd-storage-class=<etcd-storage-class> \ 5
  --ssh-key  <path-to-ssh-key> \ 6
  --namespace <hosted-cluster-namespace> \ 7
  --control-plane-availability-policy SingleReplica \
  --release-image=quay.io/openshift-release-dev/ocp-release:<ocp-release> 8
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
3. Hosted Control Plane 名前空間を指定します (例: **clusters-example**)。**oc get agent -n <hosted-control-plane-namespace>** コマンドを使用して、この名前空間でエージェントが使用可能であることを確認します。
4. ベースドメインを指定します (例: **krnl.es**)。
5. etcd ストレージクラス名を指定します (例: **lvm-storageclass**)。
6. SSH 公開鍵へのパスを指定します。デフォルトのファイルパスは **~/.ssh/id_rsa.pub** です。
7. ホストされたクラスターの名前空間を指定します。
8. 使用するサポートされている OpenShift Container Platform のバージョンを指定します (例: **4.14.0-x86_64**)。

3. しばらくしてから、次のコマンドを入力して、Hosted control plane の Pod が稼働中であることを確認します。

```
oc -n <hosted-control-plane-namespace> get pods
```

以下の出力例を参照してください。

```
NAME                                   READY   STATUS    RESTARTS   AGE
catalog-operator-6cd867cc7-phb2q       2/2     Running   0          2m50s
control-plane-operator-f6b4c8465-4k5dh   1/1     Running   0          4m32s
```

1.7.8.5.1. コンソールを使用した非ベアメタルエージェントマシンでのホステッドクラスターの作成

1. OpenShift Container Platform Web コンソールを開き、管理者の認証情報を入力してログイン

1. OpenShift Container Platform Web コンソールを開き、管理者の認証情報を入力してログインします。コンソールを開く手順については、OpenShift Container Platform ドキュメントの Web コンソールへのアクセス を参照してください。

2. コンソールヘッダーで、**All Clusters** が選択されていることを確認します。

3. **Infrastructure** > **Clusters** をクリックします。

4. **Create cluster** **Host inventory** > **Hosted control plane**をクリックします。
Create cluster ページが表示されます。

5. **Create cluster** ページでプロンプトに従い、クラスター、ノードプール、ネットワーク、および自動化に関する詳細を入力します。
注: クラスターに関する詳細を入力する際には、次のヒントが役立つ場合があります。

   - 事前定義された値を使用してコンソールのフィールドに自動的に値を入力する場合は、ホストインベントリーの認証情報を作成できます。詳細は、**オンプレミス環境の認証情報の作成** を参照してください。

   - **Cluster details** ページのプルシークレットは、OpenShift Container Platform リソースへのアクセスに使用する OpenShift Container Platform プルシークレットです。ホストインベントリー認証情報を選択した場合は、プルシークレットが自動的に入力されます。

   - **Node pools** ページでは、namespace にノードプールのホストが含まれます。コンソールを使用してホストインベントリーを作成した場合、コンソールは専用の namespace を作成します。

   - **Networking** ページで、API サーバー公開ストラテジーを選択します。ホステッドクラスターの API サーバーは、既存のロードバランサーを使用するか、**NodePort** タイプのサービスとして公開できます。API サーバーに到達できる宛先を指す **api.${HOSTED_CLUSTER_NAME}.${BASEDOMAIN}** 設定に DNS エントリーを含める必要があります。このエントリーとして、管理クラスター内のノードの 1 つを指すレコード、または受信トラフィックを Ingress Pod にリダイレクトするロードバランサーを指すレコードを指定できます。

6. エントリーを確認し、**Create** をクリックします。
Hosted cluster ビューが表示されます。

7. **Hosted cluster** ビューでホストされたクラスターのデプロイメントを監視します。ホストされたクラスターに関する情報が表示されない場合は、**All Clusters** が選択されていることを確認し、クラスター名をクリックします。コントロールプレーンコンポーネントの準備が整うまで待ちます。このプロセスには数分かかる場合があります。

8. ノードプールのステータスを表示するには、**NodePool** セクションまでスクロールします。ノードをインストールするプロセスには約 10 分かかります。**Nodes** をクリックして、ノードがホストされたクラスターに参加したかどうかを確認することもできます。

1.7.8.5.2. 関連情報

- コンソールでホステッドクラスターを作成するときに再利用できる認証情報を作成するには、オンプレミス環境の認証情報の作成 を参照してください。

- ホステッドクラスターをインポートするには、Hosted control plane クラスターの手動インポート を参照してください。

- ホステッドクラスターにアクセスするには、ホステッドクラスターへのアクセス を参照してください。

- Discovery Image を使用してホストインベントリーにホストを追加するには、Discovery Image を使用したホストインベントリーへのホストの追加 を参照してください。

1.7.8.6. ホステッドクラスター作成の確認

デプロイメントプロセスが完了したら、ホステッドクラスターが正常に作成されたことを確認できます。ホステッドクラスターの作成から数分後に、次の手順に従います。

1. 次の extract コマンドを入力して、新しいホステッドクラスターの kubeconfig を取得します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=- > kubeconfig-<hosted-cluster-name>
```

2. kubeconfig を使用して、ホステッドクラスターのクラスター Operator を表示します。以下のコマンドを入力します。

```
oc get co --kubeconfig=kubeconfig-<hosted_cluster_name>
```

以下の出力例を参照してください。

```
NAME                                       VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
console                                    4.10.26   True        False         False      2m38s
csi-snapshot-controller                    4.10.26   True        False         False      4m3s
dns                                        4.10.26   True        False         False      2m52s
```

3. 次のコマンドを入力して、ホステッドクラスター上で実行中の Pod を表示することもできます。

```
oc get pods -A --kubeconfig=kubeconfig-<hosted-cluster-name>
```

以下の出力例を参照してください。

```
NAMESPACE                                          NAME                                                READY   STATUS      RESTARTS      AGE
kube-system                                        konnectivity-agent-khlqv                            0/1     Running     0             3m52s
openshift-cluster-samples-operator                 cluster-samples-operator-6b5bcb9dff-kpnbc           2/2     Running     0             20m
openshift-monitoring                               alertmanager-main-0                                 6/6     Running     0             100s
openshift-monitoring                               openshift-state-metrics-677b9fb74f-qqp6g            3/3     Running     0             104s
```

1.7.8.7. ホステッドクラスターの NodePool オブジェクトのスケーリング

NodePool オブジェクトをスケーリングして、ホステッドクラスターにノードを追加します。

1. **NodePool** オブジェクトを 2 つのノードにスケーリングします。

```
oc -n <hosted-cluster-namespace> scale nodepool <nodepool-name> --replicas 2
```

ClusterAPI Agent エージェントプロバイダーは、ホステッドクラスターに割り当てられる 2 つ

のエージェントをランダムに選択します。これらのエージェントはさまざまな状態を経て、最終的に OpenShift Container Platform ノードとしてホステッドクラスターに参加します。エージェントは次の順序で状態を通過します。

- **binding**
- **discovering**
- **insufficient**
- **installing**
- **installing-in-progress**
- **added-to-existing-cluster**

2. 以下のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get agent
```

以下の出力例を参照してください。

```
NAME                                   CLUSTER        APPROVED   ROLE         STAGE
4dac1ab2-7dd5-4894-a220-6a3473b67ee6   hypercluster1   true       auto-assign
```

3. 以下のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get agent -o jsonpath='{range .items[*]}BMH:
{@.metadata.labels.agent-install\.openshift\.io/bmh} Agent: {@.metadata.name} State:
{@.status.debugInfo.state}{"\n"}{end}'
```

以下の出力例を参照してください。

```
BMH: ocp-worker-2 Agent: 4dac1ab2-7dd5-4894-a220-6a3473b67ee6 State: binding
BMH: ocp-worker-1 Agent: da503cf1-a347-44f2-875c-4960ddb04091 State: insufficient
```

4. 次の extract コマンドを入力して、新しいホステッドクラスターの kubeconfig を取得します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig -
-to=- > kubeconfig-<hosted-cluster-name>
```

5. エージェントが **added-to-existing-cluster** 状態に達したら、次のコマンドを入力して、ホステッドクラスターの OpenShift Container Platform ノードが表示されることを確認します。

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get nodes
```

以下の出力例を参照してください。

```
NAME          STATUS   ROLES    AGE     VERSION
ocp-worker-1   Ready    worker   5m41s   v1.24.0+3882f8f
```

Cluster Operator は、ワークロードをノードに追加することによって調整を開始します。

6. 次のコマンドを入力して、**NodePool** オブジェクトをスケールアップしたときに 2 台のマシンが作成されたことを確認します。

```
oc -n <hosted-control-plane-namespace> get machines
```

以下の出力例を参照してください。

```
NAME                          CLUSTER           NODENAME       PROVIDERID
PHASE     AGE   VERSION
hypercluster1-c96b6f675-m5vch   hypercluster1-b2qhl   ocp-worker-1   agent://da503cf1-
a347-44f2-875c-4960ddb04091   Running   15m   4.13z
```

clusterversion 調整プロセスは最終的に、Ingress および Console クラスター Operator のみが欠落しているポイントに到達します。

7. 以下のコマンドを入力します。

```
oc --kubeconfig kubeconfig-<hosted-cluster-name> get clusterversion,co
```

以下の出力例を参照してください。

```
NAME                                         VERSION   AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version             False       True          40m     Unable to apply
4.13z: the cluster operator console has not yet successfully rolled out

NAME                                                VERSION   AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                                4.13z     False       False
False      11m     RouteHealthAvailable: failed to GET route (https://console-openshift-
console.apps.hypercluster1.domain.com): Get "https://console-openshift-
console.apps.hypercluster1.domain.com": dial tcp 10.19.3.29:443: connect: connection
refused
clusteroperator.config.openshift.io/csi-snapshot-controller                4.13z     True        False
False      10m
clusteroperator.config.openshift.io/dns                                    4.13z     True        False
False      9m16s
```

1.7.8.7.1. ノードプールの追加

名前、レプリカの数、およびエージェントラベルセレクターなどの追加情報を指定して、ホステッドクラスターのノードプールを作成できます。

1. ノードプールを作成するには、次の情報を入力します。

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"

hcp create nodepool agent \
  --cluster-name $CLUSTER_NAME \
  --name $NODEPOOL_NAME \
  --node-count $WORKER_COUNT \
  --agentLabelSelector '{"matchLabels": {"size": "medium"}}' 1
```

1 **--agentLabelSelector** は任意です。ノードプールは、**"size" : "medium"** ラベルを持つエージェントを使用します。

2. **clusters** namespace 内のノードプールリソースをリストして、**nodepool** プールのステータスを確認します。

```
oc get nodepools --namespace clusters
```

3. 次のコマンドを入力して、**admin-kubeconfig** シークレットを抽出します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>-admin-kubeconfig --to=./hostedcluster-secrets --confirm
```

以下の出力例を参照してください。

```
hostedcluster-secrets/kubeconfig
```

4. しばらくしてから、次のコマンドを入力してノードプールのステータスを確認できます。

```
oc --kubeconfig ./hostedcluster-secrets get nodes
```

5. 次のコマンドを入力して、使用可能なノードプールの数が予想されるノードプールの数と一致することを確認します。

```
oc get nodepools --namespace clusters
```

1.7.8.7.2. 関連情報

- データプレーンをゼロにスケールダウンするには、データプレーンをゼロにスケールダウンする を参照してください。

1.7.8.8. 非ベアメタルエージェントマシン上のホステッドクラスターでの Ingress の処理

すべての OpenShift Container Platform クラスターには、通常、外部 DNS レコードが関連付けられているデフォルトのアプリケーション Ingress コントローラーがあります。たとえば、ベースドメイン **krnl.es** を使用して **example** という名前のホステッドクラスターを作成する場合は、ワイルドカードドメイン ***.apps.example.krnl.es** がルーティング可能であると予想できます。

***.apps** ドメインにロードバランサーとワイルドカード DNS レコードを設定するには、ゲストクラスターで次のアクションを実行します。

1. MetalLB Operator の設定が含まれる YAML ファイルを作成して、MetalLB をデプロイします。

```
apiVersion: v1
kind: Namespace
metadata:
  name: metallb
  labels:
    openshift.io/cluster-monitoring: "true"
  annotations:
    workload.openshift.io/allowed: management
---
```

```
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  name: metallb-operator-operatorgroup
  namespace: metallb
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: metallb-operator
  namespace: metallb
spec:
  channel: "stable"
  name: metallb-operator
  source: redhat-operators
  sourceNamespace: openshift-marketplace
```

2. ファイルを **metallb-operator-config.yaml** として保存します。

3. 以下のコマンドを入力して設定を適用します。

```
oc apply -f metallb-operator-config.yaml
```

4. Operator の実行後、MetalLB インスタンスを作成します。

   a. MetalLB インスタンスの設定を含む YAML ファイルを作成します。

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb
```

   b. ファイルを **metallb-instance-config.yaml** として保存します。

   c. 次のコマンドを入力して、MetalLB インスタンスを作成します。

```
oc apply -f metallb-instance-config.yaml
```

5. 2 つのリソースを作成して MetalLB Operator を設定します。

   - 単一の IP アドレスを持つ **IPAddressPool** リソース。この IP アドレスは、クラスターノードが使用するネットワークと同じサブネット上にある必要があります。

   - **IPAddressPool** リソースが BGP プロトコルを通じて提供するロードバランサーの IP アドレスをアドバタイズするための **BGP** アドバタイズリソース。

     a. 設定を含む YAML ファイルを作成します。

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: <ip_address_pool_name> 1
  namespace: metallb
spec:
```

```
  protocol: layer2
  autoAssign: false
  addresses:
    - <ingress_ip>-<ingress_ip> 2
---
apiVersion: metallb.io/v1beta1
kind: BGPAdvertisement
metadata:
  name: <bgp_advertisement_name> 3
  namespace: metallb
spec:
  ipAddressPools:
    - <ip_address_pool_name> 4
```

1 4 **IPAddressPool** リソース名を指定します。

2 環境の IP アドレスを指定します（例： **192.168.122.23** ）。

3 **BGP アドバタイズ** リソース名を指定します。

a. ファイルを **ipaddresspool-bgpadvertisement-config.yaml** として保存します。

b. 次のコマンドを入力してリソースを作成します。

```
oc apply -f ipaddresspool-bgpadvertisement-config.yaml
```

1. **LoadBalancer** タイプのサービスを作成した後、MetalLB はサービスの外部 IP アドレスを追加します。

c. **metallb-loadbalancer-service.yaml** という名前の YAML ファイルを作成して、ingress トラフィックを Ingress デプロイメントにルーティングする新しいロードバランサーサービスを設定します。

```
kind: Service
apiVersion: v1
metadata:
  annotations:
    metallb.universe.tf/address-pool: ingress-public-ip
  name: metallb-ingress
  namespace: openshift-ingress
spec:
  ports:
    - name: http
      protocol: TCP
      port: 80
      targetPort: 80
    - name: https
      protocol: TCP
      port: 443
      targetPort: 443
  selector:
    ingresscontroller.operator.openshift.io/deployment-ingresscontroller: default
  type: LoadBalancer
```

d. ファイルを **metallb-loadbalancer-service.yaml** として保存します。

e. 以下のコマンドを入力して YAML 設定を適用します。

```
oc apply -f metallb-loadbalancer-service.yaml
```

f. 次のコマンドを入力して、OpenShift Container Platform コンソールにアクセスします。

```
curl -kI https://console-openshift-console.apps.example.krnl.es

HTTP/1.1 200 OK
```

g. **clusterversion** と **clusteroperator** の値をチェックして、すべてが実行されていることを確認します。以下のコマンドを入力します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion,co
```

以下の出力例を参照してください。

```
NAME                                         VERSION         AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.x.y           True        False         3m32s   Cluster
version is 4.x.y

NAME                                                VERSION         AVAILABLE
PROGRESSING   DEGRADED   SINCE   MESSAGE
clusteroperator.config.openshift.io/console                 4.x.y           True        False
False      3m50s
clusteroperator.config.openshift.io/ingress                 4.x.y           True        False
False      53m
```

+ **4.x.y** は、使用する OpenShift Container Platform バージョン(**4.14.0-x86_64** など)に置き換えます。

1.7.8.8.1. 関連情報

- MetalLB の詳細は、OpenShift Container Platform ドキュメントの MetalLB および MetalLB Operator について を参照してください。

1.7.8.9. ホステッドクラスターのノード自動スケーリングの有効化

ホステッドクラスターにさらに容量が必要で、予備のエージェントが利用可能な場合は、自動スケーリングを有効にして新しいワーカーノードをインストールできます。

1. 自動スケーリングを有効にするには、次のコマンドを入力します。この例では、ノードの最小数は 2 で、最大数は 5 です。追加できるノードの最大数は、プラットフォームによって制限される場合があります。たとえば、エージェントプラットフォームを使用する場合、ノードの最大数は使用可能なエージェントの数によって制限されます。

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p
'[{"op": "remove", "path": "/spec/replicas"},{"op":"add", "path": "/spec/autoScaling", "value": {
"max": 5, "min": 2 }}]'
```

2. 新しいノードを必要とするワークロードを作成します。

a. 次の例で使用して、ワークロード設定を含むYAMLファイルを作成します。

a. 次の例を使用して、ワークロード設定を含む YAML ファイルを作成します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: reversewords
  name: reversewords
  namespace: default
spec:
  replicas: 40
  selector:
    matchLabels:
      app: reversewords
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: reversewords
    spec:
      containers:
      - image: quay.io/mavazque/reversewords:latest
        name: reversewords
        resources:
          requests:
            memory: 2Gi
status: {}
```

b. ファイルを **workload-config.yaml** として保存します。

c. 以下のコマンドを入力して、YAML を適用します。

```
oc apply -f workload-config.yaml
```

3. 次のコマンドを入力して、**admin-kubeconfig** シークレットを抽出します。

```
oc extract -n <hosted-cluster-namespace> secret/<hosted-cluster-name>admin-kubeconfig --
to=./hostedcluster-secrets --confirm
```

以下の出力例を参照してください。

```
hostedcluster-secrets/kubeconfig
```

4. 次のコマンドを入力して、新しいノードが **Ready** ステータスであるかどうかを確認できます。

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
```

5. ノードを削除するには、次のコマンドを入力してワークロードを削除します。

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig -n default delete deployment
reversewords
```

6. 数分間待ちます。その間に、容量の追加が必要にならないようにします。エージェントプラットフォームでは、エージェントは廃止され、再利用できます。次のコマンドを入力すると、ノードが削除されたことを確認できます。

```
oc --kubeconfig <hosted-cluster-name>.kubeconfig get nodes
```

1.7.8.9.1. ホストされたクラスターのノードの自動スケーリングを無効にする

ノードの自動スケーリングを無効にするには、次のコマンドを入力します。

```
oc -n <hosted-cluster-namespace> patch nodepool <hosted-cluster-name> --type=json -p '[\
{"op":"remove", "path": "/spec/autoScaling"}, \{"op": "add", "path": "/spec/replicas", "value": <specify-value-to-scale-replicas>]'
```

このコマンドは、YAML ファイルから **"spec.autoScaling"** を削除し、**"spec.replicas"** を追加し、**"spec.replicas"** を指定の整数値に設定します。

1.7.8.10. 非ベアメタルエージェントマシン上のホステッドクラスターの破棄

コンソールを使用して、非ベアメタルのホステッドクラスターを破棄できます。非ベアメタルエージェントマシン上のホステッドクラスターを破棄するには、次の手順を実行します。

1. コンソールで、**Infrastructure** > **Clusters** に移動します。
2. **Clusters** ページで、破棄するクラスターを選択します。
3. **Actions** メニューで **Destroy clusters** を選択し、クラスターを削除します。

1.7.8.10.1. コマンドラインを使用した非ベアメタルエージェントマシン上のホステッドクラスターの破棄

ホステッドクラスターを破棄するには、次の手順を実行します。

- 次のコマンドを実行して、ホステッドクラスターとそのバックエンドリソースを削除します。

```
hcp destroy cluster agent --name <hosted_cluster_name>
```

<hosted_cluster_name> をホストされたクラスターの名前に置き換えます。

1.7.9. 64 ビット x86 OpenShift Container Platform クラスターでのホスティングクラスターの設定による、IBM Power コンピュートノードの hosted control plane の作成 (テクノロジープレビュー)

テクノロジープレビュー: IBM Power (**ppc64le**) コンピュートノード用の 64 ビット x86 ベアメタル上でのホスティングクラスターの設定のサポートには制限があります。

ホスティングクラスターとして機能するようにクラスターを設定することで、Hosted control plane をデプロイメントできます。ホスティングクラスターは、コントロールプレーンがホストされる OpenShift Container Platform クラスターです。ホスティングクラスターは **管理** クラスターとも呼ばれます。

注:management クラスターは **マネージド** クラスターではありません。マネージドクラスターは、ハブクラスターが管理するクラスターです。

multicluster engine Operator 2.5 は、管理対象のハブクラスターであるデフォルトの **local-cluster** と、ホスティングクラスターとしてのハブクラスターのみをサポートします。

重要:

- エージェントプラットフォームを使用して、Hosted control plane をベアメタルでプロビジョニングできます。エージェントプラットフォームは、Central Infrastructure Management サービスを使用して、ホステッドクラスターにワーカーノードを追加します。中央インフラストラクチャー管理サービスの概要については、ホストインベントリーの作成 を参照してください。
- 各 IBM Power システムホストは、中央インフラストラクチャー管理が提供する Discovery イメージを使用して起動する必要があります。各ホストが起動すると、エージェントプロセスが実行されてホストの詳細が検出され、インストールが完了します。Agent カスタムリソースは、各ホストを表します。
- エージェントプラットフォームでホステッドクラスターを作成すると、HyperShift は Hosted Control Plane (HCP) namespace に Agent Cluster API プロバイダーをインストールします。
- ノードプールをスケールアップすると、マシンが作成されます。Cluster API プロバイダーは、承認され、検証に合格し、現在使用されておらず、ノードプールの仕様で指定されている要件を満たすエージェントを見つけます。エージェントのステータスと状態を確認することで、エージェントのインストールを監視できます。
- ノードプールをスケールダウンすると、エージェントは対応するクラスターからバインド解除されます。クラスターを再利用する前に、Discovery Image を使用してクラスターを再起動し、ノード数を更新する必要があります。

1.7.9.1. 前提条件

ホスティングクラスターを設定するには、次の前提条件を満たす必要があります。

- OpenShift Container Platform クラスターにインストールされた Kubernetes Operator 2.5 以降のマルチクラスターエンジン。multicluster engine Operator は、Red Hat Advanced Cluster Management をインストールすると自動的にインストールされます。OpenShift Container Platform OperatorHub から Operator として Red Hat Advanced Cluster Management を使用せずに、multicluster engine Operator をインストールすることもできます。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は、multicluster engine Operator 2.5 以降で自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

```
oc get managedclusters local-cluster
```

- HyperShift Operator を実行するには、3 つ以上のワーカーノードを含むホスティングクラスターが必要です。
- central infrastructure management サービスが有効である。詳細は、Central Infrastructure Management サービスの有効化 を参照してください。
- Hosted control plane コマンドラインインターフェイスをインストールする。Hosted control plane コマンドラインインターフェイスのインストール を参照してください。

1.7.9.2. IBM Power インフラストラクチャーの要件

エージェントプラットフォームはインフラストラクチャーを作成しませんが、インフラストラクチャーとして次のものが必要です。

- エージェント: **エージェント** は Discovery Image で起動され、OpenShift Container Platform ノードとしてプロビジョニングする準備ができているホストを表します。
- DNS: API および Ingress エンドポイントは、ルーティング可能である必要があります。

1.7.9.3. IBM Power 設定ドキュメント

前提条件を満たしたら、次のトピックを参照して、ベアメタル上に Hosted control plane を設定します。

1. InfraEnv リソースにエージェントを追加する
2. IBM Power での Hosted control plane の DNS の設定
3. ベアメタルでのホステッドクラスターの作成
4. IBM Power コンピューティングノード用の 64 ビット x86 ベアメタル上への hosted control plane 用の InfraEnv リソース作成
5. IBM Power 上のホステッドクラスターの NodePool オブジェクトのスケーリング

1.7.9.4. InfraEnv リソースにエージェントを追加する

エージェントを追加するには、ライブ ISO で開始するようにマシンを手動で設定できます。

1. ライブ ISO をダウンロードし、それを使用してホスト (ベアメタルまたは VM) を起動します。ライブ ISO の URL は、**InfraEnv** リソースの **status.isoDownloadURL** フィールドにあります。起動時に、ホストは Assisted Service と通信し、**InfraEnv** リソースと同じ namespace にエージェントとして登録します。
2. エージェントとそのプロパティーの一部を一覧表示するには、次のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get agents
```

以下の出力例を参照してください。

```
NAME                                   CLUSTER   APPROVED   ROLE          STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218                 auto-assign
e57a637f-745b-496e-971d-1abbf03341ba                 auto-assign
```

3. 各エージェントが作成された後、オプションでその **install_disk_id** と **hostname** を仕様に設定し、次のコマンドを入力してエージェントを承認できます。

```
oc -n <hosted-control-plane-namespace> patch agent 86f7ac75-4fc4-4b36-8130-40fa12602218 -p '{"spec":{"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-0.example.krnl.es"}}' --type merge

oc -n <hosted-control-plane-namespace> patch agent 23d0c614-2caa-43f5-b7d3-0b3564688baa -p '{"spec":{"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-1.example.krnl.es"}}' --type merge
```

4. エージェントの使用が承認されていることを確認するには、次のコマンドを入力して出力を確認します。

```
oc -n <hosted-control-plane-namespace> get agents
```

以下の出力例を参照してください。

```
NAME                                   CLUSTER   APPROVED   ROLE         STAGE
86f7ac75-4fc4-4b36-8130-40fa12602218             true       auto-assign
e57a637f-745b-496e-971d-1abbf03341ba             true       auto-assign
```

1.7.9.5. IBM Power での Hosted control plane の DNS の設定

ホステッドクラスターの API サーバーが公開されます。API サーバーに到達可能な宛先を指す **api.<hosted-cluster-name>.<base-domain>** エントリーの DNS エントリーが存在する必要があります。

DNS エントリーは、Hosted control plane を実行しているマネージドクラスター内のノードの 1 つを指すレコードと同様、単純化できます。

エントリーは、受信トラフィックを Ingress Pod にリダイレクトするためにデプロイされるロードバランサーを指すこともできます。

次の DNS 設定例を参照してください。

```
$ cat /var/named/<example.krnl.es.zone>
```

以下の出力例を参照してください。

```
$ TTL 900
@ IN  SOA bastion.example.krnl.es.com. hostmaster.example.krnl.es.com. (
      2019062002
      1D 1H 1W 3H )
  IN NS bastion.example.krnl.es.com.
;
;
api                IN A 1xx.2x.2xx.1xx (1)
api-int             IN A 1xx.2x.2xx.1xx
;
;
*.apps.<hosted-cluster-name>.<basedomain>          IN A 1xx.2x.2xx.1xx
;
;EOF
```

(1) このレコードは、Hosted control plane の受信トラフィックと送信トラフィックを処理する API ロードバランサーの IP アドレスを参照します。

IBM Power の場合、エージェントの IP アドレスに対応する IP アドレスを追加します。

```
compute-0            IN A 1xx.2x.2xx.1yy
compute-1            IN A 1xx.2x.2xx.1yy
```

1.7.9.6. IBM Power コンピューティングノード用の 64 ビット x86 ベアメタル上への hosted control plane 用の InfraEnv リソース作成

InfraEnv は、ライブ ISO を開始しているホストがエージェントとして参加できる環境です。この場合、エージェントは Hosted control plane と同じ namespace に作成されます。

InfraEnv リソースを作成するには、次の手順を実行します。

1. 設定を含む YAML ファイルを作成します。以下の例を参照してください。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: <hosted-cluster-name>
  namespace: <hosted-control-plane-namespace>
spec:
  cpuArchitecture: ppc64le
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <ssh-public-key>
```

2. ファイルを **infraenv-config.yaml** として保存します。

3. 次のコマンドを入力して設定を適用します。

```
oc apply -f infraenv-config.yaml
```

4. URL を取得してライブ ISO をダウンロードし、IBM Power マシンがエージェントとして参加できるようにするには、以下のコマンドを入力します。

```
oc -n <hosted-control-plane-namespace> get InfraEnv <hosted-cluster-name> -o json
```

1.7.9.7. IBM Power 上のホステッドクラスターの NodePool オブジェクトのスケーリング

NodePool オブジェクトは、ホステッドクラスターの作成時に作成されます。**NodePool** オブジェクトをスケーリングすることで、ホストされたコントロールプレーンにより多くのコンピュートノードを追加できます。

1. 次のコマンドを実行して、**NodePool** オブジェクトを 2 つのノードにスケーリングします。

```
oc -n <clusters_namespace> scale nodepool <nodepool_name> --replicas 2
```

ClusterAPI Agent エージェントプロバイダーは、ホステッドクラスターに割り当てられる 2 つのエージェントをランダムに選択します。これらのエージェントはさまざまな状態を経て、最終的に OpenShift Container Platform ノードとしてホステッドクラスターに参加します。エージェントは次の順序で移行フェーズを通過します。

- **binding**
- **discovering**
- **insufficient**
- **installing**

- **installing-in-progress**
- **added-to-existing-cluster**

2. 次のコマンドを実行して、スケールされた特定のエージェントのステータスを確認します。

```
oc -n <hosted_control_plane_namespace> get agent -o jsonpath='{range .items[*]}BMH:
{@.metadata.labels.agent-install\.openshift\.io/bmh} Agent: {@.metadata.name} State:
{@.status.debugInfo.state}{"\n"}{end}'
```

以下の出力を参照してください。

```
BMH: Agent: 50c23cda-cedc-9bbd-bcf1-9b3a5c75804d State: known-unbound
BMH: Agent: 5e498cd3-542c-e54f-0c58-ed43e28b568a State: insufficient
```

3. 次のコマンドを実行して、移行フェーズを表示します。

```
oc -n <hosted_control_plane_namespace> get agent
```

以下の出力を参照してください。

```
NAME                                   CLUSTER          APPROVED   ROLE          STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   hosted-forwarder   true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a                      true       auto-assign
da503cf1-a347-44f2-875c-4960ddb04091   hosted-forwarder   true       auto-assign
```

4. 次のコマンドを実行して、ホステッドクラスターにアクセスするための **kubeconfig** ファイルを生成します。

```
hcp create kubeconfig --namespace <clusters_namespace> --name
<hosted_cluster_namespace> > <hosted_cluster_name>.kubeconfig
```

5. エージェントが **added-to-existing-cluster** 状態に達したら、次のコマンドを入力して、OpenShift Container Platform ノードが表示されることを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get nodes
```

以下の出力を参照してください。

```
NAME                                STATUS   ROLES    AGE     VERSION
worker-zvm-0.hostedn.example.com Ready    worker   5m41s   v1.24.0+3882f8f
worker-zvm-1.hostedn.example.com Ready    worker   6m3s    v1.24.0+3882f8f
```

6. 次のコマンドを入力して、**NodePool** オブジェクトをスケールアップしたときに 2 台のマシンが作成されたことを確認します。

```
oc -n <hosted_control_plane_namespace> get machine.cluster.x-k8s.io
```

以下の出力を参照してください。

```
NAME                                CLUSTER  NODENAME PROVIDERID     PHASE     AGE
VERSION
hosted-forwarder-79558597ff-5tbqp   hosted-forwarder-crqq5   worker-zvm-
```

```
0.hostedn.example.com   agent://50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   Running   41h
4.15.0
hosted-forwarder-79558597ff-lfjfk   hosted-forwarder-crqq5   worker-zvm-
1.hostedn.example.com   agent://5e498cd3-542c-e54f-0c58-ed43e28b568a   Running   41h
4.15.0
```

7. 次のコマンドを実行して、クラスターのバージョンとクラスター Operator のステータスを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion
```

以下の出力を参照してください。

```
NAME                                         VERSION      AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.15.0   True        False         40h     Cluster
version is 4.15.0
```

8. 以下のコマンドを実行して、クラスター Operator のステータスを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusteroperators
```

クラスターの各コンポーネントの出力には、**NAME**、**VERSION**、**AVAILABLE**、**PROGRESSING**、**DEGRADED**、**SINCE**、および **MESSAGE** のクラスター Operator のステータスが表示されます。

出力例は、OpenShift Container Platform ドキュメントの 初期 Operator 設定 セクションを参照してください。

1.7.9.7.1. 関連情報

- データプレーンをゼロにスケールダウンするには、データプレーンをゼロにスケールダウンする を参照してください。

1.7.10. IBM Z コンピュートノード用の x86 ベアメタル上でのホスティングクラスターの設定 (テクノロジープレビュー)

テクノロジープレビュー: IBM Z (**s390x**) コンピュートノード用の **x86** ベアメタル上でのホスティングクラスターの設定は、テクノロジープレビュー状態であり、サポートに制限があります。

ホスティングクラスターとして機能するようにクラスターを設定することで、Hosted control plane をデプロイメントできます。ホスティングクラスターは、コントロールプレーンがホストされる OpenShift Container Platform クラスターです。ホスティングクラスターは **管理** クラスターとも呼ばれます。

注:management クラスターは **マネージド** クラスターではありません。マネージドクラスターは、ハブクラスターが管理するクラスターです。

hypershift アドオンを使用してマネージドクラスターをホスティングクラスターに変換し、そのクラスターに HyperShift Operator をデプロイできます。その後、ホステッドクラスターの作成を開始できます。

multicluster engine Operator 2.5 は、管理対象のハブクラスターであるデフォルトの **local-cluster** と、ホスティングクラスターとしてのハブクラスターのみをサポートします。

重要:

- エージェントプラットフォームを使用して、Hosted control plane をベアメタルでプロビジョニングできます。エージェントプラットフォームは、Central Infrastructure Management サービスを使用して、ホステッドクラスターにワーカーノードを追加します。central infrastructure management サービスの概要は、Kube API - Getting Started Guide を参照してください。
- 各 IBM Z システムホストは、central infrastructure management によって提供される PXE イメージを使用して起動する必要があります。各ホストが起動すると、エージェントプロセスが実行されてホストの詳細が検出され、インストールが完了します。Agent カスタムリソースは、各ホストを表します。
- エージェントプラットフォームでホステッドクラスターを作成すると、HyperShift Operator は Hosted Control Plane (HCP) namespace に Agent Cluster API プロバイダーをインストールします。
- ノードプールをスケールアップすると、マシンが作成されます。Cluster API プロバイダーは、承認され、検証に合格し、現在使用されておらず、ノードプールの仕様で指定されている要件を満たすエージェントを見つけます。エージェントのステータスと状態を確認することで、エージェントのインストールを監視できます。
- ノードプールをスケールダウンすると、エージェントは対応するクラスターからバインド解除されます。クラスターを再利用する前に、PXE イメージを使用してクラスターを起動し、ノード数を更新する必要があります。

1.7.10.1. 前提条件

- OpenShift Container Platform クラスターに Kubernetes Operator バージョン 2.5 以降のマルチクラスターエンジンをインストールする。multicluster engine Operator は、Red Hat Advanced Cluster Management をインストールすると自動的にインストールされます。OpenShift Container Platform OperatorHub から Operator として Red Hat Advanced Cluster Management を使用せずに、multicluster engine Operator をインストールすることもできます。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は、multicluster engine Operator 2.5 以降で自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

```
oc get managedclusters local-cluster
```

- HyperShift Operator を実行するために 3 つ以上のワーカーノードを含むホスティングクラスターがある。
- central infrastructure management サービスが有効である。詳細は、Central Infrastructure Management サービスの有効化 を参照してください。
- Hosted control plane コマンドラインインターフェイスをインストールする。Hosted control plane コマンドラインインターフェイスのインストール を参照してください。

1.7.10.2. IBM Z インフラストラクチャーの要件

エージェントプラットフォームはインフラストラクチャーを作成しませんが、インフラストラクチャーとして次のものが必要です。

- エージェント: **エージェント** は Discovery Image または PXE イメージで起動され、OpenShift Container Platform ノードとしてプロビジョニングする準備ができているホストを表します。
- DNS: API および Ingress エンドポイントは、ルーティング可能である必要があります。

Hosted control plane 機能がデフォルトで有効になりました。機能を無効にした後、手動で有効にする場合、または機能を無効にする必要がある場合は、Hosted control plane 機能の有効化または無効化 を参照してください。

1.7.10.3. IBM Z 設定ドキュメント

前提条件を満たしたら、次のトピックを参照して、ベアメタル上に Hosted control plane を設定します。

1. IBM Z を使用した Hosted Control Plane の DNS の設定
2. ベアメタルでのホステッドクラスターの作成
3. IBM Z コンピューティングノード用の 64 ビット x86 ベアメタル上への Hosted Control Plane 用の InfraEnv リソース作成
4. IBM Z エージェントを InfraEnv リソースに追加する (テクノロジープレビュー)
5. IBM Z 上のホステッドクラスターの NodePool オブジェクトのスケーリング

1.7.10.4. IBM Z エージェントを InfraEnv リソースに追加する (テクノロジープレビュー)

IBM Z 環境にエージェントを追加するには、追加の手順が必要です。これについては、このセクションで詳しく説明します。

注: 特に明記されていない限り、これらの手順は、IBM Z および IBM LinuxONE 上の z/VM と RHEL KVM の両方のインストールに適用されます。

1.7.10.4.1. KVM を使用した IBM Z のエージェントの追加

KVM を使用する IBM Z の場合は、次のコマンドを実行して、**InfraEnv** リソースからダウンロードした PXE イメージを使用して IBM Z 環境を開始します。エージェントが作成されると、ホストは Assisted Service と通信し、管理クラスター上の **InfraEnv** リソースと同じ namespace に登録します。

```
virt-install \
  --name "<vm_name>" \
  --autostart \
  --ram=16384 \
  --cpu host \
  --vcpus=4 \
  --location "<path_to_kernel_initrd_image>,kernel=kernel.img,initrd=initrd.img" \
  --disk <qcow_image_path> \
  --network network:macvtap-net,mac=<mac_address> \
  --graphics none \
  --noautoconsole \
  --wait=-1 \
  --extra-args "rd.neednet=1 nameserver=<nameserver>
coreos.live.rootfs_url=http://<http_server>/rootfs.img random.trust_cpu=on rd.luks.options=discard
ignition.firstboot ignition.platform.id=metal console=tty1 console=ttyS1,115200n8
coreos.inst.persistent-kargs=console=tty1 console=ttyS1,115200n8"
```

ISO ブートの場合は、**InfraEnv** リソースから ISO をダウンロードし、次のコマンドを実行してノードを起動します。

```
virt-install \
  --name "<vm_name>" \
  --autostart \
  --memory=16384 \
  --cpu host \
  --vcpus=4 \
  --network network:macvtap-net,mac=<mac_address> \
  --cdrom "<path_to_image.iso>" \
  --disk <qcow_image_path> \
  --graphics none \
  --noautoconsole \
  --os-variant <os_version> \
  --wait=-1 \
```

1.7.10.4.2. z/VM を使用した IBM のエージェントの追加

注記: z/VM ゲストに静的 IP を使用する場合は、IP パラメーターが 2 回目の起動でも持続するように、z/VM エージェントの **NMStateConfig** 属性を設定する必要があります。

InfraEnv リソースからダウンロードした PXE イメージを使用して IBM Z 環境を開始するには、以下の手順を実行します。エージェントが作成されると、ホストは Assisted Service と通信し、管理クラスター上の **InfraEnv** リソースと同じ namespace に登録します。

1. パラメーターファイルを更新して、**rootfs_url**、**network_adaptor**、および **disk_type** の値を追加します。
   以下のパラメーターファイルの例を参照してください。

```
rd.neednet=1 \
console=ttysclp0  \
coreos.live.rootfs_url=<rootfs_url> \
ip=<IP_guest_vm>::<nameserver>:255.255.255.0::<network_adaptor>:none \
nameserver=<nameserver> \
zfcp.allow_lun_scan=0 \ 1
rd.znet=qeth,<network_adaptor_range>,layer2=1 \
rd.<disk_type>=<storage> random.trust_cpu=on \ 2
rd.luks.options=discard \
ignition.firstboot ignition.platform.id=metal \
console=tty1 console=ttyS1,115200n8 \
coreos.inst.persistent-kargs="console=tty1 console=ttyS1,115200n8
```

   1. VSwitch を使用したインストールの場合は、**zfcp.allow_lun_scan=0** を追加します。OSA、Hipersockets、および RoCE を使用したインストールの場合は、このエントリーを省略します。

   2. DASD タイプのディスクにインストールする場合は、**rd.dasd=** を使用してインストールディスクを指定します。FCP タイプのディスクにインストールする場合は、**rd.zfcp=** を使用します。

2. 次のコマンドを実行して、**initrd**、カーネルイメージ、およびパラメーターファイルをゲスト VM に移動します。

```
vmur pun -r -u -N kernel.img $INSTALLERKERNELLOCATION/<image name>
```

```
vmur pun -r -u -N generic.parm $PARMFILELOCATION/paramfilename
```

```
vmur pun -r -u -N initrd.img $INSTALLERINITRAMFSLOCATION/<image name>
```

3. ゲスト VM コンソールから次のコマンドを実行します。

```
cp ipl c
```

4. エージェントとそのプロパティーをリスト表示するには、次のコマンドを入力します。

```
oc -n <hosted_control_plane_namespace> get agents
```

以下の出力例を参照してください。

```
NAME    CLUSTER APPROVED    ROLE    STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d    auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a    auto-assign
```

5. 次のコマンドを実行してエージェントを承認します。**オプション:** 仕様でエージェント ID **<installation_disk_id>** と **<hostname>** を設定できます。

```
oc -n <hosted_control_plane_namespace> patch agent 50c23cda-cedc-9bbd-bcf1-
9b3a5c75804d -p '{"spec":
{"installation_disk_id":"/dev/sda","approved":true,"hostname":"worker-zvm-
0.hostedn.example.com"}}' --type merge
```

6. 次のコマンドを実行して、エージェントが承認されていることを確認します。

```
oc -n <hosted_control_plane_namespace> get agents
```

以下の出力例を参照してください。

```
NAME                                    CLUSTER     APPROVED   ROLE         STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d          true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a          true       auto-assign
```

1.7.10.5. IBM Z を使用した Hosted control plane の DNS の設定

ホステッドクラスターの API サーバーは、NodePort サービスとして公開されます。API サーバーに到達可能な宛先を指す **api.<hosted-cluster-name>.<base-domain>** の DNS エントリーが存在する必要があります。

DNS エントリーは、Hosted control plane を実行しているマネージドクラスター内のノードの 1 つを指すレコードと同様、単純化できます。

エントリーは、受信トラフィックを Ingress Pod にリダイレクトするためにデプロイされるロードバランサーを指すこともできます。

次の DNS 設定例を参照してください。

```
$ cat /var/named/<example.krnl.es.zone>
```

以下の出力例を参照してください。

```
$ TTL 900
@ IN  SOA bastion.example.krnl.es.com. hostmaster.example.krnl.es.com. (
      2019062002
      1D 1H 1W 3H )
  IN NS bastion.example.krnl.es.com.
;
;
api                IN A 1xx.2x.2xx.1xx 1
api-int             IN A 1xx.2x.2xx.1xx
;
;
*.apps        IN A 1xx.2x.2xx.1xx
;
;EOF
```

1 このレコードは、Hosted control plane の受信トラフィックと送信トラフィックを処理する API ロードバランサーの IP アドレスを参照します。

IBM z/VM の場合、エージェントの IP アドレスに対応する IP アドレスを追加します。

```
compute-0             IN A 1xx.2x.2xx.1yy
compute-1             IN A 1xx.2x.2xx.1yy
```

1.7.10.6. IBM Z コンピューティングノードの x86 ベアメタル上への Hosted control plane 用の InfraEnv リソース作成

InfraEnv は、PXE イメージを使用して起動されるホストがエージェントとして参加できる環境です。この場合、エージェントは Hosted control plane と同じ namespace に作成されます。

InfraEnv リソースを作成するには、次の手順を参照してください。

1. 設定を含む YAML ファイルを作成します。以下の例を参照してください。

```
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: <hosted-cluster-name>
  namespace: <hosted-control-plane-namespace>
spec:
  cpuArchitecture: s390x
  pullSecretRef:
    name: pull-secret
  sshAuthorizedKey: <ssh-public-key>
```

2. ファイルを **infraenv-config.yaml** として保存します。
3. 次のコマンドを入力して設定を適用します。

```
oc apply -f infraenv-config.yaml
```

4. **initrd.img**、**kernel.img**、または **rootfs.img** などの PXE イメージをダウンロードする URL を取得するには、次のコマンドを入力します。このイメージは、IBM Z マシンがエージェントとして参加できるようにします。

```
oc -n <hosted-control-plane-namespace> get InfraEnv <hosted-cluster-name> -o json
```

1.7.10.7. IBM Z 上のホステッドクラスターの NodePool オブジェクトのスケーリング

NodePool オブジェクトは、ホステッドクラスターの作成時に作成されます。**NodePool** オブジェクトをスケーリングすることで、ホストされたコントロールプレーンにより多くのコンピュートノードを追加できます。

1. 次のコマンドを実行して、**NodePool** オブジェクトを 2 つのノードにスケーリングします。

```
oc -n <clusters_namespace> scale nodepool <nodepool_name> --replicas 2
```

ClusterAPI Agent エージェントプロバイダーは、ホステッドクラスターに割り当てられる 2 つのエージェントをランダムに選択します。これらのエージェントはさまざまな状態を経て、最終的に OpenShift Container Platform ノードとしてホステッドクラスターに参加します。エージェントは次の順序で移行フェーズを通過します。

- **binding**
- **discovering**
- **insufficient**
- **installing**
- **installing-in-progress**
- **added-to-existing-cluster**

2. 次のコマンドを実行して、スケールされた特定のエージェントのステータスを確認します。

```
oc -n <hosted_control_plane_namespace> get agent -o jsonpath='{range .items[*]}BMH:
{@.metadata.labels.agent-install\.openshift\.io/bmh} Agent: {@.metadata.name} State:
{@.status.debugInfo.state}{"\n"}{end}'
```

以下の出力を参照してください。

```
BMH: Agent: 50c23cda-cedc-9bbd-bcf1-9b3a5c75804d State: known-unbound
BMH: Agent: 5e498cd3-542c-e54f-0c58-ed43e28b568a State: insufficient
```

3. 次のコマンドを実行して、移行フェーズを表示します。

```
oc -n <hosted_control_plane_namespace> get agent
```

以下の出力を参照してください。

```
NAME                                   CLUSTER          APPROVED   ROLE          STAGE
50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   hosted-forwarder   true       auto-assign
5e498cd3-542c-e54f-0c58-ed43e28b568a                      true       auto-assign
da503cf1-a347-44f2-875c-4960ddb04091   hosted-forwarder   true       auto-assign
```

4. 次のコマンドを実行して、ホステッドクラスターにアクセスするための **kubeconfig** ファイルを生成します。

```
hcp create kubeconfig --namespace <clusters_namespace> --name
<hosted_cluster_namespace> > <hosted_cluster_name>.kubeconfig
```

5. エージェントが **added-to-existing-cluster** 状態に達したら、次のコマンドを入力して、OpenShift Container Platform ノードが表示されることを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get nodes
```

以下の出力を参照してください。

```
NAME                          STATUS   ROLES    AGE      VERSION
worker-zvm-0.hostedn.example.com Ready    worker   5m41s    v1.24.0+3882f8f
worker-zvm-1.hostedn.example.com Ready    worker   6m3s     v1.24.0+3882f8f
```

Cluster Operator は、ワークロードをノードに追加することによって調整を開始します。

6. 次のコマンドを入力して、**NodePool** オブジェクトをスケールアップしたときに 2 台のマシンが作成されたことを確認します。

```
oc -n <hosted_control_plane_namespace> get machine.cluster.x-k8s.io
```

以下の出力を参照してください。

```
NAME                           CLUSTER  NODENAME PROVIDERID     PHASE     AGE
VERSION
hosted-forwarder-79558597ff-5tbqp   hosted-forwarder-crqq5   worker-zvm-
0.hostedn.example.com   agent://50c23cda-cedc-9bbd-bcf1-9b3a5c75804d   Running   41h
4.15.0
hosted-forwarder-79558597ff-lfjfk   hosted-forwarder-crqq5   worker-zvm-
1.hostedn.example.com   agent://5e498cd3-542c-e54f-0c58-ed43e28b568a   Running   41h
4.15.0
```

7. 次のコマンドを実行して、クラスターのバージョンを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusterversion,co
```

以下の出力を参照してください。

```
NAME                                         VERSION      AVAILABLE   PROGRESSING   SINCE
STATUS
clusterversion.config.openshift.io/version   4.15.0-ec.2   True        False         40h     Cluster
version is 4.15.0-ec.2
```

8. 以下のコマンドを実行して、クラスター Operator のステータスを確認します。

```
oc --kubeconfig <hosted_cluster_name>.kubeconfig get clusteroperators
```

クラスターの各コンポーネントの出力には、**NAME**、**VERSION**、**AVAILABLE**、**PROGRESSING**、**DEGRADED**、**SINCE**、および **MESSAGE** のクラスター Operator のステータスが表示されます。

出力例は、OpenShift Container Platform ドキュメントの 初期 Operator 設定 セクションを参照してください。

1.7.10.7.1. 関連情報

- データプレーンをゼロにスケールダウンするには、データプレーンをゼロにスケールダウンする を参照してください。

1.7.11. OpenShift Virtualization での Hosted control plane クラスターの管理

Hosted control plane と Red Hat OpenShift Virtualization を使用すると、KubeVirt 仮想マシンによってホストされるワーカーノードを含む OpenShift Container Platform クラスターを作成できます。OpenShift Virtualization 上の Hosted control plane には、次のようないくつかの利点があります。

- Hosted control plane とホステッドクラスターを同じ基盤となるベアメタルインフラストラクチャーにパックすることで、リソースの使用率を向上します。
- Hosted control plane とホステッドクラスターを分離して強力な分離を実現
- ベアメタルノードのブートストラッププロセスを排除することで、クラスターのプロビジョニング時間を短縮します。
- 同じベース OpenShift Container Platform クラスターの下で多くのリリースを管理します

Hosted control plane 機能がデフォルトで有効になりました。

Hosted control plane のコマンドラインインターフェイス (**hcp**) を使用して、OpenShift Container Platform のホステッドクラスターを作成できます。ホステッドクラスターは、管理対象クラスターとして自動的にインポートされます。この自動インポート機能を無効にする場合は、multicluster engine operator へのホストクラスターの自動インポートの無効化 を参照してください。

重要:

- Hosted control plane の同じプラットフォームで、ハブクラスターとワーカーを実行します。
- 各ホステッドクラスターに、クラスター全体で一意の名前が必要です。multicluster engine Operator によってホストクラスターを管理するには、ホストクラスター名を既存のマネージドクラスターと同じにすることはできません。
- ホステッドクラスター名として **clusters** を使用しないでください。
- ホステッドクラスターは、マルチクラスターエンジンの operator 管理クラスターの namespace には作成できません。
- Hosted control plane のストレージを設定する場合は、etcd の推奨プラクティスを考慮してください。レイテンシー要件を満たすには、各コントロールプレーンノードで実行されるすべての Hosted control plane の etcd インスタンス専用の高速ストレージデバイスを使用します。LVM ストレージを使用して、ホストされた etcd Pod のローカルストレージクラスを設定できます。詳細は、OpenShift Container Platform ドキュメントの **推奨される etcd プラクティス** および **論理ボリュームマネージャーストレージを使用した永続ストレージ** を参照してください。

1.7.11.1. 前提条件

OpenShift Virtualization 上に OpenShift Container Platform クラスターを作成するには、以下の前提条件を満たす必要があります。

- **KUBECONFIG** 環境変数で指定された OpenShift Container Platform クラスター、バージョン 4.14 以降への管理者アクセスが必要です。
- OpenShift Container Platform ホスティングクラスターでは、次の DNS に示すように、ワイルドカード DNS ルートが有効になっている必要があります。

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

- OpenShift Container Platform ホスティングクラスターには、OpenShift Virtualization バージョン 4.14 以降がインストールされている必要があります。詳細は、Web コンソールを使用した OpenShift Virtualization のインストール を参照してください。
- OpenShift Container Platform ホスティングクラスターは、デフォルトの Pod ネットワーク CNI として OVNKubernetes を使用して設定する必要があります。
- OpenShift Container Platform ホスティングクラスターにはデフォルトのストレージクラスが必要です。詳細は、インストール後のストレージ設定 を参照してください。次の例は、デフォルトのストレージクラスを設定する方法を示しています。

```
oc patch storageclass ocs-storagecluster-ceph-rbd -p '{"metadata": {"annotations": {"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- **quay.io/openshift-release-dev** リポジトリーの有効なプルシークレットファイルが必要です。詳細は、ユーザーがプロビジョニングしたインフラストラクチャーを使用して x86_64 プラットフォームに OpenShift をインストールする を参照してください。
- Hosted control plane コマンドラインインターフェイスをインストールする 必要があります。
- クラスターをプロビジョニングする前に、ロードバランサーを設定する必要があります。詳細は、オプション: MetalLB の設定 を参照してください。
- ネットワークパフォーマンスを最適化するには、KubeVirt 仮想マシンをホストする OpenShift Container Platform クラスターで 9000 以上のネットワーク最大伝送単位 (MTU) を使用します。低い MTU 設定を使用すると、ネットワーク遅延とホストされる Pod のスループットに影響があります。MTU が 9000 以上の場合にのみ、ノードプールでマルチキューを有効にします。
- multicluster engine Operator には、少なくとも 1 つのマネージド OpenShift Container Platform クラスターが必要です。**local-cluster** は自動的にインポートされます。**local-cluster** の詳細については、詳細設定 を参照してください。次のコマンドを実行して、ハブクラスターの状態を確認できます。

```
oc get managedclusters local-cluster
```

1.7.11.2. ファイアウォールのポートの要件

ポートが管理クラスター、コントロールプレーン、ホストクラスター間で通信できるように、ファイアウォールとポートの要件を満たしていることを確認します。

- **kube-apiserver** サービスはデフォルトでポート 6443 で実行され、コントロールプレーンコンポーネント間の通信には ingress アクセスが必要です。
  - **NodePort** 公開ストラテジーを使用する場合は、**kube-apiserver** サービスに割り当てられたノードポートが公開されていることを確認してください。

    - MetalLB ロードバランシングを使用する場合は、ロードバランサーの IP アドレスに使用される IP 範囲への ingress アクセスを許可します。

- **NodePort** 公開ストラテジーを使用する場合は、**ignition-server** および **Oauth-server** 設定にファイアウォールルールを使用します。

- **konnectivity** エージェントは、ホステッドクラスター上で双方向通信を可能にするリバーストンネルを確立し、ポート 6443 でクラスター API サーバーアドレスへの egress アクセスを必要とします。この egress アクセスを使用すると、エージェントは **kube-apiserver** サービスにアクセスできます。

    - クラスター API サーバーのアドレスが内部 IP アドレスの場合は、ワークロードサブネットからポート 6443 の IP アドレスへのアクセスを許可します。

    - アドレスが外部 IP アドレスの場合は、ノードからその外部 IP アドレスにポート 6443 で送信できるように許可します。

- デフォルトのポート 6443 を変更する場合は、その変更を反映するようにルールを調整します。

- クラスター内で実行されるワークロードに必要なポートがすべて開いていることを確認してください。

- ファイアウォールルール、セキュリティーグループ、またはその他のアクセス制御を使用して、必要なソースだけにアクセスを制限します。必要な場合を除き、ポートを公開しないでください。

- 実稼働環境の場合は、ロードバランサーを使用して、単一の IP アドレスによるアクセスを簡素化します。

Red Hat OpenShift Virtualization 上の hosted control plane に関する関連資料については、次のドキュメントを参照してください。

- etcd および LVM ストレージの推奨事項の詳細は、推奨される etcd プラクティス および 論理ボリュームマネージャーストレージを使用した永続ストレージ を参照してください。

- 非接続環境で Red Hat OpenShift Virtualization に hosted control plane を設定するには、非接続環境でのホストされたコントロールプレーンの設定 を参照してください。

- Hosted control plane 機能を無効にするか、すでに無効にしていて手動で有効にする場合は、Hosted control plane 機能の有効化または無効化 を参照してください。

- Red Hat Ansible Automation Platform ジョブを実行してホステッドクラスターを管理するには、ホステッドクラスターで実行するための Ansible Automation Platform ジョブの設定 を参照してください。

1.7.11.3. KubeVirt プラットフォームを使用したホステッドクラスターの作成

OpenShift Container Platform 4.14 以降では、KubeVirt を使用してクラスターを作成でき、外部インフラストラクチャーを使用して作成することも可能です。KubeVirt を使用した作成プロセスの詳細は、以下をご覧ください。

- ホストされたクラスターの作成

- 外部インフラストラクチャーを使用したホステッドクラスターの作成

1.7.11.3.1. ホストされたクラスターの作成

1. ホストされたクラスターを作成するには、Hosted Control Plane コマンドラインインターフェイス **hcp** を使用します。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--etcd-storage-class=<etcd-storage-class> 6
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ワーカー数を指定します (例: **2**)。
3. プルシークレットへのパスを指定します (例: /**user/name/pullsecret**)。
4. メモリーの値を指定します (例: **6Gi**)。
5. CPU の値を指定します (例: **2**)。
6. etcd ストレージクラス名を指定します (例: **lvm-storageclass**)。

注記: --release-image フラグを使用して、特定の OpenShift Container Platform リリースでホステッドクラスターをセットアップできます。

--node-pool-replicas フラグに従って、2 つの仮想マシンワーカーレプリカを持つクラスターに対してデフォルトのノードプールが作成されます。

2. しばらくすると、次のコマンドを入力して、ホストされているコントロールプレーン Pod が実行されていることを確認できます。

```
oc -n clusters-<hosted-cluster-name> get pods
```

以下の出力例を参照してください。

```
NAME                                          READY   STATUS    RESTARTS   AGE
capi-provider-5cc7b74f47-n5gkr                1/1     Running   0          3m
catalog-operator-5f799567b7-fd6jw             2/2     Running   0          69s
certified-operators-catalog-784b9899f9-mrp6p  1/1     Running   0          66s
cluster-api-6bbc867966-l4dwl                  1/1     Running   0          66s
.
.
.
redhat-operators-catalog-9d5fd4d44-z8qqk      1/1     Running   0          66s
```

KubeVirt 仮想マシンによってサポートされるワーカーノードを含むホステッドクラスターは、通常、完全にプロビジョニングされるまでに 10 ~ 15 分かかります。

3. ホステッドクラスターのステータスを確認するには、次のコマンドを入力して、対応する **HostedCluster** リソースを確認します。

```
oc get --namespace clusters hostedclusters
```

完全にプロビジョニングされた **HostedCluster** オブジェクトを示す以下の出力例を参照してください。

```
NAMESPACE NAME VERSION KUBECONFIG PROGRESS AVAILABLE PROGRESSING MESSAGE
clusters example 4.x.0 example-admin-kubeconfig Completed True False The hosted control plane is available
```

4.x.0 を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

4. ホステッドクラスターへのアクセス の説明に従って、ホステッドクラスターにアクセスします。

1.7.11.3.2. 外部インフラストラクチャーを使用したホステッドクラスターの作成

デフォルトでは、HyperShift Operator は、ホステッドクラスターのコントロールプレーン Pod と、同じクラスター内の KubeVirt ワーカー VM の両方をホストします。外部インフラストラクチャー機能を使用すると、ワーカーノード VM をコントロールプレーン Pod とは別のクラスターに配置できます。

- **管理クラスター** は HyperShift Operator を実行し、ホステッドクラスターのコントロールプレーン Pod をホストする OpenShift Container Platform クラスターです。
- **インフラストラクチャークラスター** は、ホステッドクラスターの KubeVirt ワーカー VM を実行する OpenShift Container Platform クラスターです。
- デフォルトでは、管理クラスターは VM をホストするインフラストラクチャークラスターとしても機能します。ただし、外部インフラストラクチャーの場合、管理クラスターとインフラストラクチャークラスターは異なります。

1.7.11.3.2.1. 外部インフラストラクチャーの前提条件

- KubeVirt ノードをホストする外部インフラストラクチャークラスター上に namespace が必要です。
- 外部インフラストラクチャークラスター用の **kubeconfig** ファイルが必要です。

1.7.11.3.2.2. hcp コマンドラインインターフェイスを使用したホステッドクラスターの作成

hcp コマンドラインインターフェイスを使用して、ホステッドクラスターを作成できます。

1. KubeVirt ワーカー VM をインフラストラクチャークラスターに配置するには、次の例に示すように、**--infra-kubeconfig-file** および **--infra-namespace** 引数を使用します。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--infra-namespace=<hosted-cluster-namespace>-<hosted-cluster-name> \ 6
--infra-kubeconfig-file=<path-to-external-infra-kubeconfig> 7
```

1 ホストされているクラスターの名前を指定します (例: **example**)。

2 ワーカー数を指定します (例: **2**)。

3 プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。

4 メモリーの値を指定します (例: **6Gi**)。

5 CPU の値を指定します (例: **2**)。

6 インフラストラクチャー名前空間を指定します (例: **clusters-example**)。

7 インフラストラクチャークラスターの **kubeconfig** ファイルへのパスを指定します (例: **/user/name/external-infra-kubeconfig**)。

このコマンドを入力すると、コントロールプレーン Pod は HyperShift Operator が実行される管理クラスターでホストされ、KubeVirt VM は別のインフラストラクチャークラスターでホストされます。

2. ホステッドクラスターへのアクセス の説明に従って、ホステッドクラスターにアクセスします。

1.7.11.3.3. コンソールを使用したホステッドクラスターの作成

コンソールを使用して KubeVirt プラットフォームでホステッドクラスターを作成するには、次の手順を実行します。

1. OpenShift Container Platform Web コンソールを開き、管理者の認証情報を入力してログインします。コンソールを開く手順については、OpenShift Container Platform ドキュメントの Web コンソールへのアクセス を参照してください。
2. コンソールヘッダーで、**All Clusters** が選択されていることを確認します。
3. **Infrastructure** > **Clusters** をクリックします。
4. **Create cluster** > **Red Hat OpenShift Virtualization** > **Hosted**をクリックします。
5. **Create cluster** ページで、プロンプトに従ってクラスターとノードプールの詳細を入力します。
   注記:
   - 事前定義された値を使用してコンソールのフィールドに自動的に値を入力する場合は、Red Hat OpenShift Virtualization の認証情報を作成します。詳細は、**オンプレミス環境の認証情報の作成** を参照してください。
   - **Cluster details** ページのプルシークレットは、OpenShift Container Platform リソースへのアクセスに使用する OpenShift Container Platform プルシークレットです。Red Hat OpenShift Virtualization の認証情報を選択した場合、プルシークレットが自動的に入力されます。
6. エントリーを確認し、**Create** をクリックします。
   Hosted cluster ビューが表示されます。
7. **Hosted cluster** ビューでホストされたクラスターのデプロイメントを監視します。ホストされたクラスターに関する情報が表示されない場合は、**All Clusters** が選択されていることを確認し、クラスター名をクリックします。

8. コントロールプレーンコンポーネントの準備が整うまで待ちます。このプロセスには数分かかる場合があります。

9. ノードプールのステータスを表示するには、**NodePool** セクションまでスクロールします。ノードをインストールするプロセスには約 10 分かかります。**Nodes** をクリックして、ノードがホストされたクラスターに参加したかどうかを確認することもできます。

1.7.11.3.4. 関連情報

- コンソールでホステッドクラスターを作成するときに再利用できる認証情報を作成するには、オンプレミス環境の認証情報の作成 を参照してください。
- ホステッドクラスターにアクセスするには、ホステッドクラスターへのアクセス を参照してください。

1.7.11.4. デフォルトの Ingress と DNS の動作

すべての OpenShift Container Platform クラスターにはデフォルトのアプリケーション Ingress コントローラーが含まれており、これにはワイルドカード DNS レコードが関連付けられている必要があります。デフォルトでは、HyperShift KubeVirt プロバイダーを使用して作成されたホステッドクラスターは、自動的に KubeVirt 仮想マシンが実行される OpenShift Container Platform クラスターのサブドメインになります。

たとえば、OpenShift Container Platform クラスターには次のデフォルトの Ingress DNS エントリーがある可能性があります。

```
*.apps.mgmt-cluster.example.com
```

その結果、**guest** という名前が付けられ、その基礎となる OpenShift Container Platform クラスター上で実行される KubeVirt ホステッドクラスターには、次のデフォルト Ingress が設定されます。

```
*.apps.guest.apps.mgmt-cluster.example.com
```

デフォルトの Ingress DNS が適切に機能するには、KubeVirt 仮想マシンをホストするクラスターでワイルドカード DNS ルートを許可する必要があります。この動作は、以下のコマンドを入力して設定できます。

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

注: デフォルトのホステッドクラスター Ingress を使用する場合は、接続はポート 443 経由の HTTPS トラフィックに制限されます。ポート 80 経由のプレーン HTTP トラフィックは拒否されます。この制限は、デフォルトの Ingress の動作にのみ適用されます。

1.7.11.4.1. Ingress と DNS の動作のカスタマイズ

デフォルトの Ingress および DNS 動作を使用しない場合は、作成時に一意のベースドメインを使用して KubeVirt ホスト型クラスターを設定できます。このオプションでは、作成時に手動の設定手順が必要であり、クラスターの作成、ロードバランサーの作成、およびワイルドカード DNS 設定の 3 つの主要な手順が含まれます。

1.7.11.4.1.1. 基本ドメインを指定するホステッドクラスターのデプロイ

1. 基本ドメインを指定するホステッドクラスターを作成するには、次のコマンドを入力します。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--base-domain <basedomain> 6
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ワーカー数を指定します (例: **2**)。
3. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
4. メモリーの値を指定します (例: **6Gi**)。
5. CPU の値を指定します (例: **2**)。
6. ベースドメインを指定します (例: **hypershift.lab**)。

その結果、ホストされたクラスターには、クラスター名とベースドメイン用に設定された Ingress ワイルドカード (例: **.apps.example.hypershift.lab**) が含まれます。ホストされたクラスターは **Partial** ステータスのままです。一意のベースドメインを持つホストされたクラスターを作成した後、必要な DNS レコードとロードバランサーを設定する必要があるためです。

1. 次のコマンドを入力して、ホストされたクラスターのステータスを表示します。

```
oc get --namespace clusters hostedclusters
```

以下の出力例を参照してください。

```
NAME          VERSION   KUBECONFIG                  PROGRESS   AVAILABLE
PROGRESSING   MESSAGE
example                 example-admin-kubeconfig    Partial    True       False        The
hosted control plane is available
```

2. 次のコマンドを入力してクラスターにアクセスします。

```
hcp create kubeconfig --name <hosted-cluster-name> > <hosted-cluster-name>-kubeconfig
```

```
oc --kubeconfig <hosted-cluster-name>-kubeconfig get co
```

以下の出力例を参照してください。

```
NAME                                 VERSION   AVAILABLE   PROGRESSING   DEGRADED
SINCE   MESSAGE
console                              4.x.0     False       False         False      30m
RouteHealthAvailable: failed to GET route (https://console-openshift-
console.apps.example.hypershift.lab): Get "https://console-openshift-
console.apps.example.hypershift.lab": dial tcp: lookup console-openshift-
console.apps.example.hypershift.lab on 172.31.0.10:53: no such host
ingress                              4.x.0     True        False         True       28m     The "default"
ingress controller reports Degraded=True: DegradedConditions: One or more other status
```

```
conditions indicate a degraded state: CanaryChecksSucceeding=False
(CanaryChecksRepetitiveFailures: Canary route checks for the default ingress controller are
failing)
```

4.x.0 を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

次の手順では、出力内のエラーを修正します。

注記: ホストされたクラスターがベアメタル上にある場合は、ロードバランサーサービスを設定するために MetalLB が必要になる場合があります。詳細は、**オプション: MetalLB の設定** を参照してください。

1.7.11.4.1.2. ロードバランサーのセットアップ

Ingress トラフィックを KubeVirt 仮想マシンにルーティングし、ロードバランサー IP アドレスにワイルドカード DNS エントリーを割り当てるロードバランサーサービスを設定します。

1. ホストされたクラスターの Ingress を公開する **NodePort** サービスがすでに存在します。ノードポートをエクスポートし、それらのポートをターゲットとするロードバランサーサービスを作成できます。

   a. 次のコマンドを入力して、HTTP ノードポートを取得します。

   ```
   oc --kubeconfig <hosted-cluster-name>-kubeconfig get services -n openshift-ingress
   router-nodeport-default -o jsonpath='{.spec.ports[?(@.name=="http")].nodePort}'
   ```

   次の手順で使用する HTTP ノードポート値をメモします。

   b. 次のコマンドを入力して、HTTPS ノードポートを取得します。

   ```
   oc --kubeconfig <hosted-cluster-name>-kubeconfig get services -n openshift-ingress
   router-nodeport-default -o jsonpath='{.spec.ports[?(@.name=="https")].nodePort}'
   ```

   次の手順で使用する HTTPS ノードポート値をメモします。

2. 次のコマンドを入力して、ロードバランサーサービスを作成します。

```
oc apply -f -
apiVersion: v1
kind: Service
metadata:
  labels:
    app: <hosted-cluster-name>
  name: <hosted-cluster-name>-apps
  namespace: clusters-<hosted-cluster-name>
spec:
  ports:
  - name: https-443
    port: 443
    protocol: TCP
    targetPort: <https-node-port> 1
  - name: http-80
    port: 80
    protocol: TCP
```

```
    targetPort: <http-node-port> 2
  selector:
    kubevirt.io: virt-launcher
  type: LoadBalancer
```

1 前の手順でメモした HTTPS ノードポート値を指定します。

2 前の手順でメモした HTTP ノードポート値を指定します。

1.7.11.4.1.3. ワイルドカード DNS の設定

ロードバランサーサービスの外部 IP を参照するワイルドカード DNS レコードまたは CNAME を設定します。

1. 次のコマンドを入力して外部 IP アドレスを取得します。

```
oc -n clusters-<hosted-cluster-name> get service <hosted-cluster-name>-apps -o
jsonpath='{.status.loadBalancer.ingress[0].ip}'
```

以下の出力例を参照してください。

```
192.168.20.30
```

2. 外部 IP アドレスを参照するワイルドカード DNS エントリーを設定します。次の DNS エントリーの例を表示します。

```
*.apps.<hosted-cluster-name\>.<base-domain\>.
```

DNS エントリーは、クラスターの内部と外部にルーティングできる必要があります。次の DNS 解決の例を参照してください。

```
dig +short test.apps.example.hypershift.lab

192.168.20.30
```

3. 次のコマンドを実行して、ホストされたクラスターのステータスが **Partial** から **Completed** に移行したことを確認します。

```
oc get --namespace clusters hostedclusters
```

以下の出力例を参照してください。

```
NAME            VERSION   KUBECONFIG                  PROGRESS    AVAILABLE
PROGRESSING   MESSAGE
example         4.x.0     example-admin-kubeconfig    Completed   True        False         The
hosted control plane is available
```

4.x.0 を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

1.7.11.4.1.4. 関連情報

- OpenShift Virtualization での Hosted control plane クラスターの管理
- オプション: MetalLB の設定
- このトピックの最初の デフォルトの Ingress と DNS の動作 に戻ります。

1.7.11.5. オプション: MetalLB の設定

MetalLB を設定する前に、MetalLB Operator をインストールする必要があります。詳細は、OpenShift Container Platform ドキュメント **の MetalLB Operator のインストール** を参照してください。

ゲストクラスターで MetalLB を設定するには、次の手順を実行します。

1. 次のサンプル YAML コンテンツを **configure-metallb.yaml** ファイルに保存して、**MetalLB** リソースを作成します。

```
apiVersion: metallb.io/v1beta1
kind: MetalLB
metadata:
  name: metallb
  namespace: metallb-system
```

2. 次のコマンドを入力して、YAML コンテンツを適用します。

```
oc apply -f configure-metallb.yaml
```

以下の出力例を参照してください。

```
metallb.metallb.io/metallb created
```

3. 以下のサンプル YAML コンテンツを **create-ip-address-pool.yaml** ファイルに保存して、**IPAddressPool** リソースを作成します。

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: metallb
  namespace: metallb-system
spec:
  addresses:
  - 192.168.216.32-192.168.216.122 1
```

1 ノードネットワーク内で使用可能な IP アドレスの範囲を使用してアドレスプールを作成します。IP アドレス範囲は、ネットワーク内で使用可能な IP アドレスの未使用のプールに置き換えます。

4. 次のコマンドを入力して、YAML コンテンツを適用します。

```
oc apply -f create-ip-address-pool.yaml
```

以下の出力例を参照してください。

```
ipaddresspool.metallb.io/metallb created
```

5. 次のサンプル YAML コンテンツを **l2advertisement.yaml** ファイルに保存して、**L2Advertisement** リソースを作成します。

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
   - metallb
```

6. 次のコマンドを入力して、YAML コンテンツを適用します。

```
oc apply -f l2advertisement.yaml
```

以下の出力例を参照してください。

```
l2advertisement.metallb.io/metallb created
```

1.7.11.5.1. 関連情報

- MetalLB の詳細は、MetalLB Operator のインストール を参照してください。

1.7.11.6. 追加のネットワーク、Guaranteed CPU、およびノードプールの仮想マシンのスケジュールを設定する

ノードプール用に追加のネットワークを設定する必要がある場合、仮想マシン (VM) 用の Guaranteed CPU へのアクセスを要求する場合、または KubeVirt 仮想マシンのスケジュールを管理する必要がある場合は、次の手順を参照してください。

1.7.11.6.1. ノードプールへの複数のネットワークの追加

デフォルトでは、ノードプールによって生成されたノードは、Pod ネットワークに割り当てられます。Multus および NetworkAttachmentDefinitions を使用すると、追加のネットワークをノードに割り当てることができます。

1. 複数のネットワークをノードに追加するには、次の例に示すように **--additional-network** 引数を使用します。

```
shell linenums="1"
export CLUSTER_NAME=example
export PULL_SECRET="$HOME/pull-secret"
export MEM="6Gi"
export CPU="2"
export WORKER_COUNT="2"


hcp create cluster kubevirt \
--name $CLUSTER_NAME \
--node-pool-replicas $WORKER_COUNT \
--pull-secret $PULL_SECRET \
--memory $MEM \
```

```
--cores $CPU \
–additional-network name:<namespace/name> \ 1
–additional-network name:<namespace/name>
```

1 **–additional-network** 引数の値を **name:<namespace/name>** に設定します。**<namespace/name>** は、NetworkAttachmentDefinition の namespace と名前に置き換えます。

1.7.11.6.2. Guaranteed CPU リソースの要求

デフォルトでは、KubeVirt 仮想マシンはノード上の他のワークロードと CPU を共有する場合があります。これにより、仮想マシンのパフォーマンスに影響が出る可能性があります。パフォーマンスへの影響を回避するために、仮想マシン用の Guaranteed CPU へのアクセスを要求できます。

1. Guaranteed CPU リソースを要求するには、次の例に示すように、**--qos-class** 引数を **Delivered** に設定します。

```
shell linenums="1"
export CLUSTER_NAME=example
export PULL_SECRET="$HOME/pull-secret"
export MEM="6Gi"
export CPU="2"
export WORKER_COUNT="2"

hcp create cluster kubevirt \
--name $CLUSTER_NAME \
--node-pool-replicas $WORKER_COUNT \
--pull-secret $PULL_SECRET \
--memory $MEM \
--cores $CPU \
--qos-class Guaranteed
```

1.7.11.6.3. ノードセットに KubeVirt 仮想マシンをスケジュールする

デフォルトでは、ノードプールによって作成された KubeVirt 仮想マシンは、使用可能な任意のノードにスケジュールされます。KubeVirt 仮想マシンは、仮想マシンを実行するのに十分な容量を持つ特定のノードセットにスケジュールすることもできます。

1. ノードプール内の KubeVirt 仮想マシンを特定のノードセットにスケジュールするには、次の例に示すように **--vm-node-selector** 引数を使用します。

```
shell linenums="1"
export CLUSTER_NAME=example
export PULL_SECRET="$HOME/pull-secret"
export MEM="6Gi"
export CPU="2"
export WORKER_COUNT="2"

hcp create cluster kubevirt \
--name $CLUSTER_NAME \
--node-pool-replicas $WORKER_COUNT \
--pull-secret $PULL_SECRET \
```

```
--memory $MEM \
--cores $CPU \
--vm-node-selector labelKey1=labelVal1,labeKey2=labelVal2
```

1.7.11.7. ノードプールのスケーリング

1. **oc scale** コマンドを使用して、ノードプールを手動でスケーリングできます。

```
NODEPOOL_NAME=${CLUSTER_NAME}-work
NODEPOOL_REPLICAS=5

oc scale nodepool/$NODEPOOL_NAME --namespace clusters --
replicas=$NODEPOOL_REPLICAS
```

2. しばらくしてから、次のコマンドを入力して、ノードプールのステータスを確認します。

```
oc --kubeconfig $CLUSTER_NAME-kubeconfig get nodes
```

以下の出力例を参照してください。

```
NAME              STATUS   ROLES    AGE     VERSION
example-9jvnf       Ready    worker   97s     v1.27.4+18eadca
example-n6prw       Ready    worker   116m    v1.27.4+18eadca
example-nc6g4       Ready    worker   117m    v1.27.4+18eadca
example-thp29       Ready    worker   4m17s   v1.27.4+18eadca
example-twxns       Ready    worker   88s     v1.27.4+18eadca
```

1.7.11.7.1. ノードプールの追加

名前、レプリカの数、およびメモリーや CPU 要件などの追加情報を指定して、ホステッドクラスターのノードプールを作成できます。

1. ノードプールを作成するには、次の情報を入力します。この例では、ノードプールには VM に割り当てられたより多くの CPU があります。

```
export NODEPOOL_NAME=${CLUSTER_NAME}-extra-cpu
export WORKER_COUNT="2"
export MEM="6Gi"
export CPU="4"
export DISK="16"

hcp create nodepool kubevirt \
  --cluster-name $CLUSTER_NAME \
  --name $NODEPOOL_NAME \
  --node-count $WORKER_COUNT \
  --memory $MEM \
  --cores $CPU
  --root-volume-size $DISK
```

2. **clusters** namespace 内のノードプールリソースをリストして、**nodepool** プールのステータスを確認します。

```
oc get nodepools --namespace clusters
```

以下の出力例を参照してください。

```
NAME                  CLUSTER        DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION   UPDATINGVERSION   UPDATINGCONFIG
MESSAGE
example               example        5               5               False         False        4.x.0
example-extra-cpu         example         2                       False         False                True
True            Minimum availability requires 2 replicas, current 0 available
```

4.x.0 を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

3. しばらくしてから、次のコマンドを入力してノードプールのステータスを確認できます。

```
oc --kubeconfig $CLUSTER_NAME-kubeconfig get nodes
```

以下の出力例を参照してください。

```
NAME                  STATUS   ROLES    AGE     VERSION
example-9jvnf           Ready    worker   97s     v1.27.4+18eadca
example-n6prw           Ready    worker   116m    v1.27.4+18eadca
example-nc6g4           Ready    worker   117m    v1.27.4+18eadca
example-thp29           Ready    worker   4m17s   v1.27.4+18eadca
example-twxns           Ready    worker   88s     v1.27.4+18eadca
example-extra-cpu-zh9l5   Ready    worker   2m6s    v1.27.4+18eadca
example-extra-cpu-zr8mj   Ready    worker   102s    v1.27.4+18eadca
```

4. 次のコマンドを入力して、ノードプールが予期したステータスになっていることを確認します。

```
oc get nodepools --namespace clusters
```

以下の出力例を参照してください。

```
NAME                  CLUSTER        DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION   UPDATINGVERSION   UPDATINGCONFIG
MESSAGE
example               example        5               5               False         False        4.x.0
example-extra-cpu         example         2               2               False         False        4.x.0
```

4.x.0 を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

1.7.11.7.1.1. 関連情報

- OpenShift Virtualization での Hosted control plane クラスターの管理
- このトピックの最初の ノードプールのスケーリング に戻ります。
- データプレーンをゼロにスケールダウンするには、データプレーンをゼロにスケールダウンする を参照してください。

1.7.11.8. OpenShift Virtualization でのホステッドクラスターの作成の検証

ホステッドクラスターが正常に作成されたことを確認するには、次の手順を完了します。

1. 次のコマンドを入力して、**HostedCluster** リソースが **completed** 状態に移行したことを確認します。

```
oc get --namespace clusters hostedclusters ${CLUSTER_NAME}
```

以下の出力例を参照してください。

```
NAMESPACE NAME VERSION KUBECONFIG PROGRESS AVAILABLE PROGRESSING MESSAGE
clusters example 4.12.2 example-admin-kubeconfig Completed True False The hosted control plane is available
```

2. 次のコマンドを入力して、ホステッドクラスター内のすべてのクラスターオペレーターがオンラインであることを確認します。

```
hcp create kubeconfig --name $CLUSTER_NAME > $CLUSTER_NAME-kubeconfig
```

```
oc get co --kubeconfig=$CLUSTER_NAME-kubeconfig
```

以下の出力例を参照してください。

```
NAME VERSION AVAILABLE PROGRESSING DEGRADED SINCE MESSAGE
console 4.12.2 True False False 2m38s
csi-snapshot-controller 4.12.2 True False False 4m3s
dns 4.12.2 True False False 2m52s
image-registry 4.12.2 True False False 2m8s
ingress 4.12.2 True False False 22m
kube-apiserver 4.12.2 True False False 23m
kube-controller-manager 4.12.2 True False False 23m
kube-scheduler 4.12.2 True False False 23m
kube-storage-version-migrator 4.12.2 True False False 4m52s
monitoring 4.12.2 True False False 69s
network 4.12.2 True False False 4m3s
node-tuning 4.12.2 True False False 2m22s
openshift-apiserver 4.12.2 True False False 23m
openshift-controller-manager 4.12.2 True False False 23m
openshift-samples 4.12.2 True False False 2m15s
operator-lifecycle-manager 4.12.2 True False False 22m
operator-lifecycle-manager-catalog 4.12.2 True False False 23m
operator-lifecycle-manager-packageserver 4.12.2 True False False 23m
service-ca 4.12.2 True False False 4m41s
storage 4.12.2 True False False 4m43s
```

1.7.11.9. OpenShift Virtualization での Hosted control plane のストレージの設定

高度な設定が提供されていない場合、デフォルトのストレージクラスが KubeVirt 仮想マシン (VM) イメージ、KubeVirt CSI マッピング、および etcd ボリュームに使用されます。

1.7.11.9.1. KubeVirt CSI ストレージクラスのマッピング

KubeVirt CSI では、**ReadWriteMany** アクセスモードを持つインフラストラクチャーストレージクラスをホステッドクラスターに公開することができます。**--infra-storage-class-mapping** 引数を使用して、クラスターの作成時にインフラストラクチャークラスターストレージクラスとホストクラスターストレージクラスのマッピングを設定できます。

インフラストラクチャーストレージクラスをホステッドストレージクラスにマップするには、次の例を参照してください。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--infra-storage-class-mapping=<storage-class>/<hosted-storage-class> \ 6
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ワーカー数を指定します (例: **2**)。
3. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
4. メモリーの値を指定します (例: **6Gi**)。
5. CPU の値を指定します (例: **2**)。
6. **<storage-class>** をインフラストラクチャーストレージクラス名に置き換え、**<hosted-storage-class>** をホストされたクラスターストレージクラス名に置き換えます。**create** コマンド内で **--infra-storage-class-mapping** 引数を複数回使用できます。

ホスト型クラスターを作成すると、インフラストラクチャーストレージクラスがホストされたクラスター内に表示されます。これらのストレージクラスのいずれかを使用するホステッドクラスター内に PVC を作成すると、KubeVirt CSI はクラスターの作成時に設定したインフラストラクチャーストレージクラスマッピングを使用してそのボリュームをプロビジョニングします。

注記: KubeVirt CSI は、 **ReadWriteMany** (RWX) アクセスが可能なインフラストラクチャーストレージクラスのマッピングのみをサポートします。

1.7.11.9.2. KubeVirt VM ルートボリュームの設定

クラスターの作成時に、**--root-volume-storage-class** 引数を使用して、KubeVirt 仮想マシンルートボリュームをホストするために使用されるストレージクラスを設定できます。

KubeVirt VM のカスタムストレージクラスとボリュームサイズを設定するには、次の例を参照してください。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
```

```
--cores <value-for-cpu> \ 5
--root-volume-storage-class <root-volume-storage-class> \ 6
--root-volume-size <volume-size> 7
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ワーカー数を指定します (例: **2**)。
3. プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。
4. メモリーの値を指定します (例: **6Gi**)。
5. CPU の値を指定します (例: **2**)。
6. KubeVirt 仮想マシンルートボリュームをホストするために使用されるストレージクラスの名前を指定します (例: **ocs-storagecluster-ceph-rbd**)。
7. ボリュームサイズを指定します (例: **64**)。

結果は、**ocs-storagecluster-ceph-rdb** ストレージクラスにホストされる PVC 上でホストされる仮想マシンを含むホストされたクラスターになります。

1.7.11.9.3. KubeVirt VM イメージキャッシュの有効化

KubeVirt イメージキャッシュは、クラスターの起動時間とストレージ使用率の両方を最適化するために使用できる高度な機能です。この機能には、スマートクローン作成と **ReadWriteMany** アクセスモードが可能なストレージクラスの使用が必要です。スマートクローン作成の詳細は、**smart-cloning を使用したデータボリュームのクローン作成** を参照してください。

イメージのキャッシュは次のように機能します。

1. VM イメージは、ホステッドクラスターに関連付けられた PVC にインポートされます。
2. その PVC の一意のクローンは、クラスターにワーカーノードとして追加されるすべての KubeVirt VM に対して作成されます。

イメージキャッシュを使用すると、イメージのインポートが 1 つだけ必要になるため、VM の起動時間が短縮されます。ストレージクラスがコピーオンライトクローン作成をサポートしている場合、クラスター全体のストレージ使用量をさらに削減できます。

イメージキャッシュを有効にするには、クラスターの作成時に、次の例に示すように、**--root-volume-cache-strategy=PVC** 引数を使用します。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--root-volume-cache-strategy=PVC 6
```

1. ホストされているクラスターの名前を指定します (例: **example**)。
2. ワーカー数を指定します (例: **2**)。

3 プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。

4 メモリーの値を指定します (例: **6Gi**)。

5 CPU の値を指定します (例: **2**)。

6 イメージキャッシュのストラテジー (例: **PVC**) を指定します。

1.7.11.9.4. etcd ストレージの設定

クラスターの作成時に、**--etcd-storage-class** 引数を使用して、etcd データをホストするために使用されるストレージクラスを設定できます。

etcd のストレージクラスを設定するには、次の例を参照してください。

```
hcp create cluster kubevirt \
--name <hosted-cluster-name> \ 1
--node-pool-replicas <worker-count> \ 2
--pull-secret <path-to-pull-secret> \ 3
--memory <value-for-memory> \ 4
--cores <value-for-cpu> \ 5
--etcd-storage-class=<etcd-storage-class-name> 6
```

1 ホストされているクラスターの名前を指定します (例: **example**)。

2 ワーカー数を指定します (例: **2**)。

3 プルシークレットへのパスを指定します (例: **/user/name/pullsecret**)。

4 メモリーの値を指定します (例: **6Gi**)。

5 CPU の値を指定します (例: **2**)。

6 etcd ストレージクラス名を指定します (例: **lvm-storageclass**)。**--etcd-storage-class** 引数を指定しない場合は、デフォルトのストレージクラスが使用されます。

1.7.11.9.4.1. 関連情報

- smart-cloning を使用したデータボリュームのクローン作成

1.7.11.10. OpenShift Virtualization 上のホステッドクラスターの破棄

ホステッドクラスターとそのマネージドクラスターリソースを破棄するには、次の手順を実行します。

1. 次のコマンドを実行して、multicluster engine Operator のマネージドクラスターリソースを削除します。

   ```
   oc delete managedcluster <managed_cluster_name>
   ```

   ここで、**<managed_cluster_name>** は管理対象クラスターの名前です。

2. 次のコマンドを実行して、ホステッドクラスターとそのバックエンドリソースを削除します。

```
hcp destroy cluster kubevirt --name <hosted_cluster_name>
```

<hosted_cluster_name> をホストされたクラスター名に置き換えます。

1.7.12. 非接続環境での Hosted control plane の設定

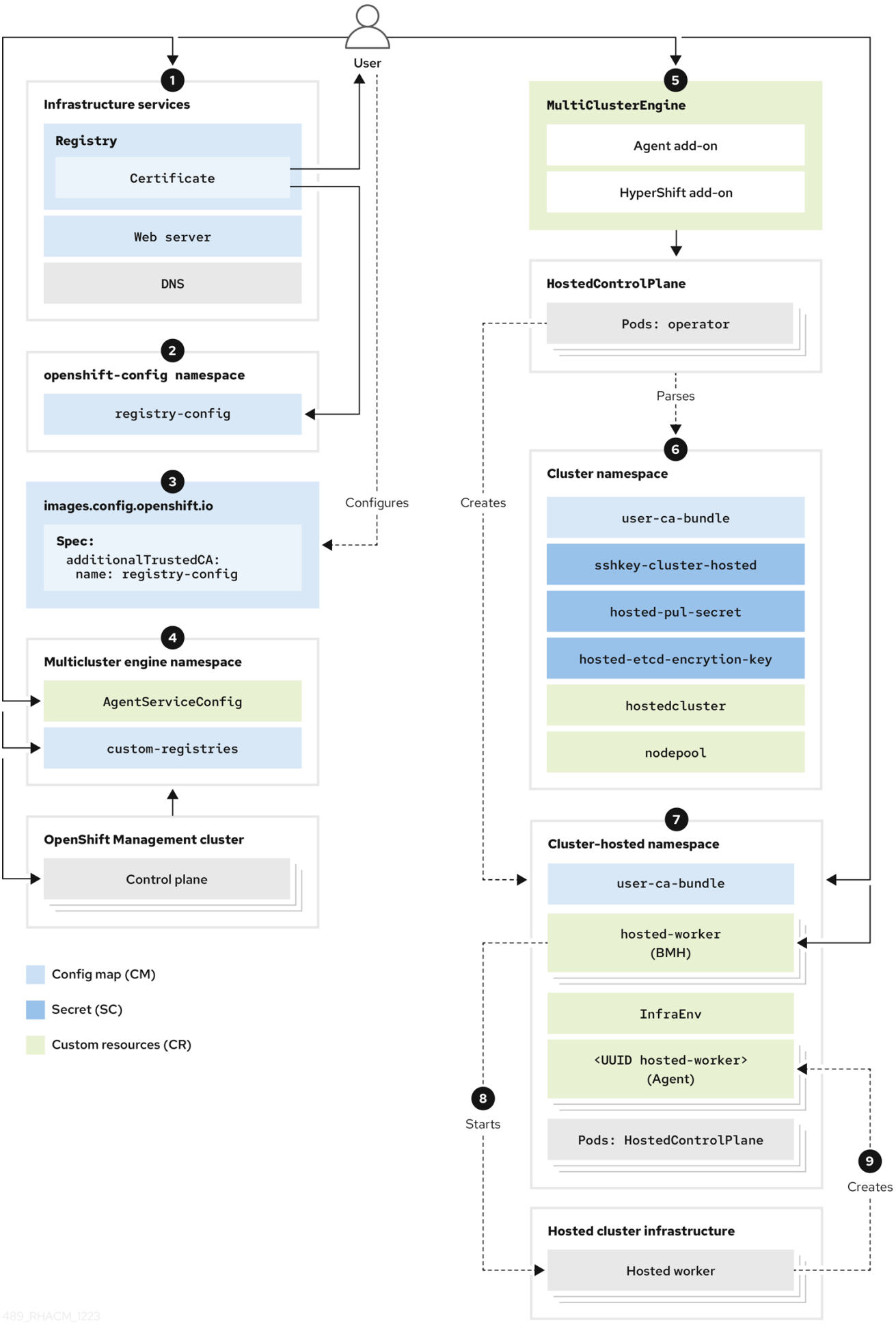
非接続環境は、インターネットに接続されておらず、Hosted Control Plane をベースとして使用する OpenShift Container Platform クラスターです。

テクノロジープレビュー: IPv4 または IPv6 ネットワークを使用して、ベアメタルプラットフォーム上の非接続環境に Hosted control plane をデプロイメントできます。さらに、非接続環境の Hosted control plane は、テクノロジープレビュー機能としてデュアルスタックネットワークで使用できます。Red Hat OpenShift Virtualization プラットフォームを使用する場合は、オフライン環境の Hosted control plane がテクノロジープレビュー機能として利用できます。

1.7.12.1. 非接続環境のアーキテクチャー

Agent プラットフォームを使用して、ベアメタル上に Hosted Control Plane をプロビジョニングできます。エージェントプラットフォームは、Central Infrastructure Management サービスを使用して、ホステッドクラスターにワーカーノードを追加します。central infrastructure management の概要は、central infrastructure management の有効化 を参照してください。

非接続環境の次のアーキテクチャー図を参照してください。


User
1
Infrastructure services
Registry
Certificate
Web server
DNS
2
openshift-config namespace
registry-config
3
images.config.openshift.io
Spec:
additionalTrustedCA:
name: registry-config
4
Multicluster engine namespace
AgentServiceConfig
custom-registries
OpenShift Management cluster
Control plane
Configures
Creates
5
MultiClusterEngine
Agent add-on
HyperShift add-on
HostedControlPlane
Pods: operator
Parses
6
Cluster namespace
user-ca-bundle
sshkey-cluster-hosted
hosted-pul-secret
hosted-etcd-encrytion-key
hostedcluster
nodepool
7
Cluster-hosted namespace
user-ca-bundle
hosted-worker
(BMH)
InfraEnv
<UUID hosted-worker>
(Agent)
Pods: HostedControlPlane
8
Starts
9
Creates
Hosted cluster infrastructure
Hosted worker
Config map (CM)
Secret (SC)
Custom resources (CR)
489_RHACM_1223

1. TLS サポートを備えたレジストリー証明書のデプロイメント、Web サーバー、DNS などのインフラストラクチャーサービスを設定して、切断されたデプロイメントが確実に機能するようにします。

2. **openshift-config** namespace に config map を作成します。config map の内容はレジストリー CA 証明書です。config map の data フィールドには、次のキーと値が含まれている必要があります。

   - キー: **<registry_dns_domain_name>..<port>** (例: **registry.hypershiftdomain.lab..5000:**) ポートを指定するときは、レジストリー DNS ドメイン名の後に **..** を配置するようにしてください。

   - 値: 証明書の内容

   config map の作成に関する詳細は、**IPv4 ネットワークの TLS 証明書の設定** を参照してください。

3. **images.config.openshift.io** カスタムリソース (CR) に、 **name: registry-config** という値を持つ **additionalTrustedCA** フィールドを追加します。

4. **multicluster-engine** namespace に config map を作成します。次のサンプル設定を参照してください。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: | 1
    -----BEGIN CERTIFICATE-----
    ...
    ...
    ...
    -----END CERTIFICATE-----
  registries.conf: | 2
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.ocp-edge-cluster-0.qe.lab.redhat.com:5000/openshift4"

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
...
...
```

1 レジストリーの CA 証明書が含まれます。

2 **registries.conf** ファイルの内容が **RAW** 形式で含まれています。

1. multicluster engine Operator namespace では、**multiclusterengine** CR を作成します。これにより、Agent アドオンと **hypershift-addon** アドオンの両方が有効になります。切断されたデプロイメントでの動作を変更するには、multicluster engine Operator namespace に config map が含まれている必要があります。namespace には、**multicluster-engine**、**assisted-service**、および **hypershift-addon-manager** Pod も含まれます。

2. ホストされたクラスターをデプロイするには、次のコンポーネントのオブジェクトを作成します。

   - シークレット: シークレットには、プルシークレット、SSH キー、etcd 暗号化キーが含まれます。
   - config map: config map には、プライベートレジストリーの CA 証明書が含まれています。
   - **HostedCluster**: **HostedCluster** リソースは、ホストされたクラスターの設定を定義します。
   - **NodePool**: **NodePool** リソースは、データプレーンに使用するマシンを参照するノードプールを識別します。

3. ホストされたクラスターオブジェクトを作成すると、HyperShift Operator は **HostedControlPlane** namespace にコントロールプレーン Pod を作成します。**HostedControlPlane** namespace は、Agent、ベアメタルホスト、**InfraEnv** リソースなどのコンポーネントもホストします。

4. **InfraEnv** リソースを作成します。ISO イメージを生成した後、ベースボード管理コントローラー (BMC) の認証情報を含むベアメタルホストとそのシークレットを作成します。

5. **openshift-machine-api** namespace の Metal3 Operator は、新しいベアメタルホストを検査します。

6. Metal3 Operator は、**LiveISO** 値および **RootFS** 値を使用して BMC に接続し、起動しようとします。マルチクラスターエンジン Operator の namespace の **AgentServiceConfig** CR を通じて、**LiveISO** 値と **RootFS** 値を指定できます。

7. **HostedCluster** リソースのワーカーノードが起動された後、エージェントコンテナーが起動されます。

8. **NodePool** リソースを、**HostedCluster** リソースのワーカーノードの数に合わせてスケーリングします。

9. デプロイメントプロセスが完了するまで待ちます。

1.7.12.2. 前提条件

オフライン環境で Hosted control plane を設定するには、次の前提条件を満たす必要があります。

- CPU: 提供される CPU の数によって、同時に実行できるホストクラスターの数が決まります。通常、3 つのノードの場合、各ノードに 16 個の CPU を使用します。最小限の開発では、3 つのノードの場合、各ノードに 12 個の CPU を使用できます。
- メモリー: RAM の量は、ホストできるホストクラスターの数に影響します。各ノードに 48 GB の RAM を使用します。最小限の開発であれば、18 GB の RAM で十分です。

- ストレージ: multicluster engine operator には SSD ストレージを使用します。
  - 管理クラスター: 250 GB。
  - レジストリー: 必要なレジストリーストレージは、ホストされるリリース、Operator、およびイメージの数によって異なります。500 GB が必要になる場合があります。ホストされたクラスターをホストするディスクとは別にすることを推奨します。
  - Web サーバー: 必要な Web サーバーストレージは、ホストされる ISO とイメージの数によって異なります。500 GB 必要になる場合があります。
- 実稼働環境: 実稼働環境の場合、管理クラスター、レジストリー、および Web サーバーを異なるディスク上に分離します。本番環境では次の設定例を参照してください。
  - レジストリー: 2TB
  - 管理クラスター: 500 GB
  - Web サーバー:2TB

1.7.12.3. OpenShift Container Platform リリースイメージダイジェストの抽出

タグ付けされたイメージを使用して、OpenShift Container Platform リリースイメージダイジェストを抽出できます。以下の手順を実行します。

1. 次のコマンドを実行して、イメージダイジェストを取得します。

```
oc adm release info <tagged_openshift_release_image> | grep "Pull From"
```

<tagged_openshift_release_image> を、サポートされている OpenShift Container Platform バージョンのタグ付きイメージに置き換えます (例: **quay.io/openshift-release-dev/ocp-release:4.14.0-x8_64**)。

以下の出力例を参照してください。

```
Pull From: quay.io/openshift-release-dev/ocp-release@sha256:69d1292f64a2b67227c5592c1a7d499c7d00376e498634ff8e1946bc9ccdddfe
```

イメージタグとダイジェストの詳細は、OpenShift Container Platform ドキュメントの **イメージストリームでのイメージの参照** を参照してください。

1.7.12.3.1. 関連情報

- IPv4 ネットワークの TLS 証明書を設定する
- イメージストリームでのイメージの参照

1.7.12.4. オフライン環境でのユーザーワークロードの監視

hypershift-addon マネージドクラスターアドオンは、HyperShift Operator の **--enable-uwm-telemetry-remote-write** オプションを有効にします。このオプションを有効にすると、ユーザーワークロードの監視が有効になり、コントロールプレーンから Telemetry メトリックをリモートで書き込むことができるようになります。

インターネットに接続されていない OpenShift Container Platform クラスターに multicluster engine Operator をインストールした場合、次のコマンドを入力して HyperShift Operator のユーザーワークロードモニタリング機能を実行しようとすると、この機能はエラーで失敗します。

```
oc get events -n hypershift
```

エラーの例を以下に示します。

```
LAST SEEN   TYPE      REASON          OBJECT               MESSAGE
4m46s       Warning   ReconcileError  deployment/operator  Failed to ensure UWM telemetry remote write: cannot get telemeter client secret: Secret "telemeter-client" not found
```

このエラーを回避するには、**local-cluster** namespace に config map を作成して、ユーザーワークロード監視オプションを無効にする必要があります。アドオンを有効にする前または後に config map を作成できます。アドオンエージェントは、HyperShift Operator を再設定します。

次の config map を作成します。

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: hypershift-operator-install-flags
  namespace: local-cluster
data:
  installFlagsToAdd: ""
  installFlagsToRemove: "--enable-uwm-telemetry-remote-write"
```

1.7.12.4.1. Hosted control plane 機能のステータス確認

Hosted control plane 機能がデフォルトで有効になりました。

1. この機能が無効になっており、有効にする場合は、次のコマンドを入力します。**multiclusterengine** は、multicluster engine Operator インスタンスの名前に置き換えます。

   ```
   oc patch mce <multiclusterengine> --type=merge -p '{"spec":{"overrides":{"components":[{"name":"hypershift","enabled": true}]}}}'
   ```

   この機能を有効にすると、**hypershift-addon** マネージドクラスターアドオンが **local-cluster** マネージドクラスターにインストールされ、アドオンエージェントは multicluster engine Operator ハブクラスターに HyperShift Operator をインストールします。

2. 次のコマンドを入力して、**hypershift-addon** マネージドクラスターアドオンがインストールされていることを確認します。

   ```
   oc get managedclusteraddons -n local-cluster hypershift-addon
   ```

3. 結果の出力を確認します。

   ```
   NAME              AVAILABLE   DEGRADED   PROGRESSING
   hypershift-addon  True        False
   ```

4. このプロセス時のタイムアウトを回避するには、以下のコマンドを入力します。

```
oc wait --for=condition=Degraded=True managedclusteraddons/hypershift-addon -n local-cluster --timeout=5m
```

```
oc wait --for=condition=Available=True managedclusteraddons/hypershift-addon -n local-cluster --timeout=5m
```

プロセスが完了すると、**hypershift-addon** マネージドクラスターアドオンと HyperShift Operator がインストールされ、**local-cluster** マネージドクラスターがホステッドクラスターをホストおよび管理できるようになります。

1.7.12.4.2. インフラストラクチャーノード上で実行する hypershift-addon マネージドクラスターアドオンの設定

デフォルトでは、**hypershift-addon** マネージドクラスターアドオンに対してノード配置設定は指定されていません。インフラストラクチャーノード上でアドオンを実行することを検討してください。そうすることで、サブスクリプション数に対する請求コストの発生や、個別のメンテナンスおよび管理タスクの発生を防ぐことができます。

1. ハブクラスターにログインします。
2. 次のコマンドを入力して、**hypershift-addon-deploy-config** アドオンデプロイメント設定仕様を開いて編集します。

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

3. 以下の例のように、**nodePlacement** フィールドを仕様に追加します。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  nodePlacement:
    nodeSelector:
      node-role.kubernetes.io/infra: ""
    tolerations:
    - effect: NoSchedule
      key: node-role.kubernetes.io/infra
      operator: Exists
```

4. 変更を保存します。**hypershift-addon** マネージドクラスターアドオンは、新規および既存のマネージドクラスターのインフラストラクチャーノードにデプロイされます。

1.7.12.5. IPv4 を使用して非接続環境で Hosted Control Plane を設定する

IPv4 ネットワークを使用して、非接続環境で Hosted Control Plane を設定できます。IPv4 範囲では、IPv6 またはデュアルスタック設定よりも必要な外部コンポーネントが少なくなります。

IPv4 ネットワーク上で Hosted Control Plane を設定するには、次の手順を確認してください。

1. IPv4 ネットワーク用のハイパーバイザーを設定する
2. IPv4 ネットワークの DNS を設定する

3. IPv4 ネットワーク用のレジストリーをデプロイする
4. IPv4 ネットワークの管理クラスターを設定する
5. IPv4 ネットワーク用の Web サーバーを設定する
6. IPv4 ネットワークのイメージミラーリングを設定する
7. IPv4 ネットワーク用の multicluster engine Operator をデプロイする
8. IPv4 ネットワークの TLS 証明書を設定する
9. IPv4 ネットワークのホステッドクラスターをデプロイする
10. IPv4 ネットワークのデプロイメントを終了する

1.7.12.5.1. IPv4 ネットワーク用のハイパーバイザーを設定する

以下の情報は、仮想マシン環境にのみ適用されます。

1.7.12.5.1.1. 仮想 OpenShift Container Platform クラスターのパッケージへのアクセスおよびデプロイ

1. 仮想 OpenShift Container Platform 管理クラスターをデプロイするには、以下のコマンドを入力して必要なパッケージにアクセスします。

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 次のコマンドを入力して、Podman サービスを有効にして起動します。

```
systemctl enable --now podman
```

3. **kcli** を使用して OpenShift Container Platform 管理クラスターおよびその他の仮想コンポーネントをデプロイするには、以下のコマンドを入力してハイパーバイザーをインストールおよび設定します。

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd
```

```
sudo dnf -y copr enable karmab/kcli
```

```
sudo dnf -y install kcli
```

```
sudo kcli create pool -p /var/lib/libvirt/images default
```

```
kcli create host kvm -H 127.0.0.1 local
```

```
sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images
```

```
kcli create network  -c 192.168.122.0/24 default
```

1.7.12.5.1.2. ネットワークマネージャーディスパッチャーの有効化

1. ネットワークマネージャーのディスパッチャーを有効にして、仮想マシンが必要なドメイン、ルート、およびレジストリーを解決できるようにします。ネットワークマネージャーディスパッチャーを有効にするには、**/etc/NetworkManager/dispatcher.d/** ディレクトリーに次の内容を含む **Forcens** という名前のスクリプトを作成し、環境に合わせて必要に応じて値を置き換えます。

```
#!/bin/bash

export IP="192.168.126.1" 1
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE 2
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

1. OpenShift Container Platform 管理クラスターをホストするハイパーバイザーインターフェイスの IP アドレスを指すように **IP** 変数を変更します。
2. **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

2. ファイルを作成したら、次のコマンドを入力してパーミッションを追加します。

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. スクリプトを実行し、出力が **ok** を返すことを確認します。

1.7.12.5.1.3. BMC アクセスの設定

1. 仮想マシンのベースボード管理コントローラー (BMC) をシミュレートするように **ksushy** を設定します。次のコマンドを入力します。

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
```

```
kcli create sushy-service --ssl --port 9000
```

```
sudo systemctl daemon-reload
```

```
systemctl enable --now ksushy
```

2. 次のコマンドを入力して、サービスが正しく機能しているかどうかをテストします。

```
systemctl status ksushy
```

1.7.12.5.1.4. 接続を許可するためのハイパーバイザーシステムの設定

開発環境で作業している場合は、環境内の各種仮想ネットワークを介したさまざまなタイプの接続を許可するようにハイパーバイザーシステムを設定します。

注: 実稼働環境で作業している場合は、安全な環境を維持するために、**firewalld** サービスの適切なルールを確立し、SELinux ポリシーを設定する必要があります。

- SELinux の場合は、次のコマンドを入力します。

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- **firewalld** の場合は、次のコマンドを入力します。

```
systemctl disable --now firewalld
```

- **libvirtd** の場合は、以下のコマンドを入力します。

```
systemctl restart libvirtd
```

```
systemctl enable --now libvirtd
```

次に、環境に合わせて DNS を設定します。

1.7.12.5.1.5. 関連情報

- **kcli**, の詳細は、公式の kcli ドキュメント を参照してください。

1.7.12.5.2. IPv4 ネットワークの DNS を設定する

この手順は、仮想環境とベアメタル環境の両方で、オフライン環境とオンライン環境の両方で必須です。仮想環境とベアメタル環境の主な違いは、リソースを設定する場所にあります。ベアメタル環境では、**dnsmasq** のような軽量のソリューションではなく、Bind のようなソリューションを使用してください。

- 仮想環境で IPv4 ネットワークの DNS を設定するには、デフォルトの Ingress と DNS の動作 を参照してください。
- ベアメタル上で IPv4 ネットワークの DNS を設定するには、ベアメタル上での DNS の設定 を参照してください。

次にレジストリーをデプロイします。

1.7.12.5.3. IPv4 ネットワーク用のレジストリーをデプロイする

開発環境の場合は、Podman コンテナーを使用して、小規模な自己ホスト型レジストリーをデプロイします。実稼働環境では、Red Hat Quay、Nexus、Artifactory などのエンタープライズでホストされるレジストリーを使用します。

Podman を使用して小規模なレジストリーをデプロイするには、以下の手順を実行します。

1. 特権ユーザーとして **${HOME}** ディレクトリーにアクセスし、次のスクリプトを作成します。

```
#!/usr/bin/env bash

set -euo pipefail

PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" 1

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi

dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -x509 -
days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
```

```
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER $REGISTRY_PASSWORD
podman create --name registry --net host --security-opt label=disable --replace -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e "REGISTRY_AUTH=htpasswd"
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -e
"REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1 **PULL_SECRET** の場所は、設定に適した場所に置き換えます。

2. スクリプトファイル **registry.sh** という名前を指定して保存します。スクリプトを実行すると、以下の情報がプルされます。
   - ハイパーバイザーのホスト名に基づくレジストリー名
   - 必要な認証情報およびユーザーアクセスの詳細
3. 次のように実行フラグを追加して、パーミッションを調整します。

```
chmod u+x ${HOME}/registry.sh
```

4. パラメーターを指定せずにスクリプトを実行するには、以下のコマンドを入力します。

```
${HOME}/registry.sh
```

   このスクリプトはサーバーを起動します。

5. このスクリプトは、管理目的で **systemd** サービスを使用します。スクリプトを管理する必要がある場合は、以下のコマンドを使用できます。

```
systemctl status
```

```
systemctl start
```

```
systemctl stop
```

レジストリーのルートフォルダーは /**opt/registry** ディレクトリー内にあり、次のサブディレクトリーが含まれています。

- **certs** には TLS 証明書が含まれます。
- **auth** には認証情報が含まれます。
- **data** にはレジストリーイメージが含まれます。
- **conf** にはレジストリー設定が含まれています。

1.7.12.5.4. IPv4 ネットワークの管理クラスターの設定

OpenShift Container Platform 管理クラスターを設定するには、dev-scripts を使用できます。または、仮想マシンをベースにしている場合は、**kcli** ツールを使用できます。以下は、**kcli** ツールに固有のものです。

1. ハイパーバイザーで使用するために適切なネットワークの準備が完了していることを確認します。ネットワークは、管理クラスターとホステッドクラスターの両方をホストします。以下の **kcli** コマンドを入力します。

   ```
   kcli create network -c 192.168.125.0/24 -P dhcp=false -P dns=false --domain
   dns.base.domain.name ipv4
   ```

   ここでは、以下のようになります。

   - **-c** は、ネットワークの CIDR を指定します。
   - **-p dhcp=false** は、設定した **dnsmasq** によって処理される DHCP を無効にするようにネットワークを設定します。
   - **-P dns=false は、** DNS を無効にするようにネットワークを設定します。これも、設定した **dnsmasq** によって処理されます。
   - **--domain** は、検索するドメインを設定します。
   - **dns.base.domain.name** は DNS ベースドメイン名です。
   - **ipv4** は、作成するネットワークの名前です。

2. ネットワークを作成したら、以下の出力を確認します。

   ```
   [root@hypershiftbm ~]# kcli list network
   Listing Networks...
   +---------+--------+--------------------+-------+------------------+------+
   | Network |  Type  |        Cidr        |  Dhcp |      Domain      | Mode |
   +---------+--------+--------------------+-------+------------------+------+
   | default | routed |   192.168.122.0/24  |  True |     default      | nat  |
   | ipv4    | routed |   192.168.125.0/24  | False | dns.base.domain.name | nat  |
   | ipv6    | routed | 2620:52:0:1306::/64 | False | dns.base.domain.name | nat  |
   +---------+--------+--------------------+-------+------------------+------+
   ```

   ```
   [root@hypershiftbm ~]# kcli info network ipv4
   Providing information about network ipv4...
   cidr: 192.168.125.0/24
   dhcp: false
   domain: dns.base.domain.name
   mode: nat
   plan: kvirt
   type: routed
   ```

3. OpenShift Container Platform 管理クラスターをデプロイできるように、プルシークレットと **kcli** プランファイルが配置されていることを確認します。

   a. プルシークレットが **kcli** プランと同じフォルダーにあり、プルシークレットファイルの名前が **openshift_pull.json** であることを確認します。

b. OpenShift Container Platform 定義を含む **kcli** プランを **mgmt-compact-hub-ipv4.yaml** ファイルに追加します。ご使用の環境に合わせてファイルの内容を更新してください。

```
plan: hub-ipv4
force: true
version: nightly
tag: "4.x.y-x86_64" (1)
cluster: "hub-ipv4"
domain: dns.base.domain.name
api_ip: 192.168.125.10
ingress_ip: 192.168.125.11
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: false
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: ipv4
users_dev: developer
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: ipv4-virtual-network
metallb_ranges:
- 192.168.125.150-192.168.125.190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:10
- hub-ctlplane-0:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:01
```

```
- hub-ctlplane-1:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:02
- hub-ctlplane-2:
    nets:
    - name: ipv4
      mac: aa:aa:aa:aa:02:03
```

1 **4.x.y** を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

4. 管理クラスターをプロビジョニングするには、以下のコマンドを入力します。

```
kcli create cluster openshift --pf mgmt-compact-hub-ipv4.yaml
```

1.7.12.5.4.1. 関連情報

- **kcli** プランファイルのパラメーターの詳細は、**kcli** 公式ドキュメントの parameters.yml の作成 を参照してください。

1.7.12.5.5. IPv4 ネットワーク用の Web サーバーを設定する

ホステッドクラスターとしてデプロイする OpenShift Container Platform リリースに関連付けられた Red Hat Enterprise Linux CoreOS (RHCOS) イメージをホストするには、追加の Web サーバーを設定する必要があります。

Web サーバーを設定するには、以下の手順を実行します。

1. 以下のコマンドを入力して、使用する OpenShift Container Platform リリースから **openshift-install** バイナリーを展開します。

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}"
```

2. 次のスクリプトを実行します。このスクリプトは、**/opt/srv** ディレクトリーにフォルダーを作成します。このフォルダーには、ワーカーノードをプロビジョニングするための RHCOS イメージが含まれています。

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*
```

```
## Run Webserver
podman ps --noheading | grep -q websrv-ai
if [[ $? == 0 ]];then
    echo "Launching Registry pod..."
    /usr/bin/podman run --name websrv-ai --net host -v /opt/srv:/usr/local/apache2/htdocs:z
quay.io/alosadag/httpd:p8080
fi
```

1. **ROOTFS_IMG_URL** 値は OpenShift CI Release ページにあります。
2. **LIVE_ISO_URL** 値は、OpenShift CI リリースページで確認できます。

ダウンロードが完了すると、コンテナーが実行され、Web サーバー上でイメージをホストします。このコンテナーは公式 HTTPd イメージのバリエーションを使用しているので、IPv6 ネットワークでの動作も可能になります。

1.7.12.5.6. IPv4 ネットワークのイメージミラーリングを設定する

イメージミラーリングは、**registry.redhat.com** や **quay.io** などの外部レジストリーからイメージを取得し、プライベートレジストリーに保存するプロセスです。

1.7.12.5.6.1. ミラーリングプロセスの完了

注: ミラーリングプロセスは、レジストリーサーバーの実行後に開始してください。

次の手順では、**ImageSetConfiguration** オブジェクトを使用するバイナリーである、**oc-mirror** ツールが使用されます。このファイルで、以下の情報を指定できます。

- ミラーリングする OpenShift Container Platform バージョン。バージョンは **quay.io** にあります。
- ミラーリングする追加の Operator。パッケージは個別に選択します。
- リポジトリーに追加する追加のイメージ。

イメージのミラーリングを設定するには、以下の手順を実行します。

1. **${HOME}/.docker/config.json** ファイルが、ミラーリング元のレジストリーとイメージのプッシュ先のプライベートレジストリーで更新されていることを確認します。
2. 次の例を使用して、ミラーリングに使用する **ImageSetConfiguration** オブジェクトを作成します。環境に合わせて必要に応じて値を置き換えます。

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest 1
mirror:
  platform:
    channels:
    - name: candidate-4.14
      minVersion: 4.x.y-x86_64 2
      maxVersion: 4.x.y-x86_64
      type: ocp
```

```
  graph: true
additionalImages:
- name: quay.io/karmab/origin-keepalived-ipfailover:latest
- name: quay.io/karmab/kubectl:latest
- name: quay.io/karmab/haproxy:latest
- name: quay.io/karmab/mdns-publisher:latest
- name: quay.io/karmab/origin-coredns:latest
- name: quay.io/karmab/curl:latest
- name: quay.io/karmab/kcli:latest
- name: quay.io/user-name/trbsht:latest
- name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
operators:
- catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
  packages:
  - name: lvms-operator
  - name: local-storage-operator
  - name: odf-csi-addons-operator
  - name: odf-operator
  - name: mcg-operator
  - name: ocs-operator
  - name: metallb-operator
  - name: kubevirt-hyperconverged
```

1. **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。
2. **4.x.y** を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

3. 次のコマンドを入力して、ミラーリングプロセスを開始します。

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

ミラーリングプロセスが完了すると、**oc-mirror-workspace/results-XXXXXX/** という名前の新しいフォルダーが作成されます。このフォルダーには、ICSP と、ホステッドクラスターに適用するカタログソースが含まれます。

4. **oc adm release Mirror** コマンドを使用して、OpenShift Container Platform の夜間バージョンまたは CI バージョンをミラーリングします。以下のコマンドを入力します。

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
  --from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
  --to=${REGISTRY}/openshift/release \
  --to-release-image=${REGISTRY}/openshift/release-images:4.x.y-x86_64
```

4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

5. 非接続ネットワークへのインストール の手順に従って、最新の multicluster engine Operator イメージをミラーリングします。

1.7.12.5.6.2. 管理クラスターでのオブジェクトの適用

ミラーリングプロセスが完了したら、管理クラスターに 2 つのオブジェクトを適用する必要があります。

- イメージコンテンツソースポリシー (ICSP) またはイメージダイジェストミラーセット (IDMS)
- カタログソース

oc-mirror ツールを使用すると、出力アーティファクトは **oc-mirror-workspace/results-XXXXXX/** という名前のフォルダーに保存されます。

ICSP または IDMS は、ノードを再起動せずに、各ノードで kubelet を再起動する **MachineConfig** 変更を開始します。ノードが **READY** としてマークされたら、新しく生成されたカタログソースを適用する必要があります。

カタログソースは、カタログイメージのダウンロードや処理を行い、そのイメージに含まれるすべての **packagemanifests** を取得するなど、**openshift-marketplace** Operator でアクションを開始します。

1. 新しいソースを確認するには、新しい **CatalogSource** をソースとして使用して次のコマンドを実行します。

```
oc get packagemanifest
```

2. アーティファクトを適用するには、次の手順を実行します。
   a. 次のコマンドを入力して、ImageContentSourcePolicy (ICSP) または IDMS アーティファクトを作成します。

```
oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
```

   b. ノードの準備が完了するまで待ってから、次のコマンドを入力します。

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

3. OLM カタログをミラーリングし、ホステッドクラスターがミラーを指すように設定します。**管理** (デフォルト) OLMCatalogPlacement モードを使用する場合、OLM カタログに使用されるイメージストリームは、管理クラスター上の ICSP からのオーバーライド情報で自動的に修正されません。
   a. OLM カタログが元の名前とタグを使用して内部レジストリーに適切にミラーリングされている場合は、**hypershift.openshift.io/olm-catalogs-is-registry-overrides** アノテーションを **HostedCluster** リソースに追加します。形式は **"sr1=dr1,sr2=dr2** " です。ソースレジストリーの文字列はキーで、宛先のレジストリーは値になります。
   b. OLM カタログイメージストリームメカニズムをバイパスするには、**HostedCluster** リソースで次の 4 つのアノテーションを使用して、OLM Operator カタログに使用する 4 つのイメージのアドレスを直接指定します。
      - **hypershift.openshift.io/certified-operators-catalog-image**
      - **hypershift.openshift.io/community-operators-catalog-image**
      - **hypershift.openshift.io/redhat-marketplace-catalog-image**
      - **hypershift.openshift.io/redhat-operators-catalog-image**

この場合、イメージストリームは作成されないため、Operator の更新を取り込むために中

この場合、イメージストリームは作成されないため、Operator の更新を取り込むために内部ミラーの更新時に、アノテーションの値を更新する必要があります。

注: 上書きメカニズムが必要な場合は、4 つのデフォルトのカタログソースの値 4 つすべてが必要です。

1.7.12.5.6.3. 関連情報

- 仮想環境で作業している場合は、ミラーリングを設定した後、OpenShift Virtualization 上のホストされたコントロールプレーンの前提条件 を満たしていることを確認してください。
- OpenShift Container Platform の夜間ミラーリングまたは CI バージョンのミラーリングの詳細は、oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング を参照してください。

1.7.12.5.7. IPv4 ネットワーク用の multicluster engine Operator をデプロイする

multicluster engine Operator は、プロバイダー間でクラスターをデプロイメントする場合に重要な役割を果たします。Red Hat Advanced Cluster Management をすでにインストールしている場合は、multicluster engine Operator は自動的にインストールされるため、インストールする必要はありません。

multicluster engine Operator がインストールされていない場合は、次のドキュメントを参照して、前提条件とインストール手順を確認してください。

- multicluster engine operator を使用したクラスターライフサイクルについて
- multicluster engine operator のインストールとアップグレード

1.7.12.5.7.1. AgentServiceConfig リソースのデプロイ

AgentServiceConfig カスタムリソースは、multicluster engine operator の一部である Assisted Service アドオンの重要なコンポーネントです。このコンポーネントは、ベアメタルクラスターをデプロイメントします。アドオンが有効な場合に、**AgentServiceConfig** リソースをデプロイしてアドオンを設定します。

AgentServiceConfig リソースの設定に加えて、multicluster engine Operator が非接続環境で適切に機能するように、追加の config map を含める必要があります。

1. 次の config map を追加してカスタムレジストリーを設定します。これには、デプロイメントをカスタマイズするための非接続環境の情報が含まれています。

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]
```

```
    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" (1)

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

(1) **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

オブジェクトには、以下の 2 つのフィールドが含まれます。

- カスタム CA: このフィールドには、デプロイメントのさまざまなプロセスに読み込まれる認証局 (CA) が含まれます。
- レジストリー: **Registries.conf** フィールドには、元のソースレジストリーではなくミラーレジストリーから使用する必要があるイメージと namespace に関する情報が含まれています。

2. 次の例に示すように、**AssistedServiceConfig** オブジェクトを追加して、Assisted Service を設定します。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  annotations:
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-service-config
(1)
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries (2)
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
```

```
      storage: 20Gi
  osImages: 3
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img 4
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-0-
live.x86_64.iso
    version: 414.92.202308281054-0
```

1. **metadata.annotations"unsupported.agent-install.openshift.io/assisted-service-configmap"** アノテーションは、Operator が動作をカスタマイズするために使用する config map 名を参照します。
2. **spec.mirrorRegistryRef.name** アノテーションは、Assisted Service Operator が使用する非接続のレジストリー情報を含む config map を指します。この config map は、デプロイメントプロセス中にこれらのリソースを追加します。
3. **spec.osImages** フィールドには、この Operator がデプロイできるさまざまなバージョンが含まれています。このフィールドは必須です。この例では、**RootFS** ファイルと **LiveISO** ファイルがすでにダウンロードされていることを前提としています。
4. **rootFSUrl フィールド** と **url** フィールドで、**dns.base.domain.name** を DNS ベースドメイン名に置き換えます。

3. すべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに適用し、これらのオブジェクトをデプロイします。起動するには、以下のコマンドを実行します。

```
oc apply -f agentServiceConfig.yaml
```

このコマンドは、次の出力例に示すように、2 つの Pod をトリガーします。

```
assisted-image-service-0                  1/1   Running   2          11d 1
assisted-service-668b49548-9m7xw          2/2   Running   5          11d 2
```

1. **Assisted-image-service** Pod は、デプロイするクラスターごとにカスタマイズされた、Red Hat Enterprise Linux CoreOS (RHCOS) 起動イメージテンプレートを作成します。
2. **assisted-service** は Operator を参照します。

1.7.12.5.8. IPv4 ネットワークの TLS 証明書を設定する

オフライン環境で Hosted control plane を設定するプロセスに、いくつかの TLS 証明書が関与します。認証局 (CA) を管理クラスターに追加するには、OpenShift Container Platform コントロールプレーンおよびワーカーノード内の以下のファイルの内容を変更する必要があります。

- **/etc/pki/ca-trust/extracted/pem/**
- **/etc/pki/ca-trust/source/anchors**
- **/etc/pki/tls/certs/**

CA を管理クラスターに追加するには、次の手順を実行します。

1. OpenShift Container Platform の公式ドキュメントの CA バンドルの更新 の手順を完了します。この方法には、CA を OpenShift Container Platform ノードにデプロイする **image-registry-operator** の使用が含まれます。

2. この方法が実際の状況に該当しない場合は、管理クラスター内の **openshift-config** namespace に **user-ca-bundle** という名前の config map が含まれているかどうかを確認してください。

   - namespace にその config map が含まれている場合は、次のコマンドを入力します。

     ```
     ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
     export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

     oc create configmap user-ca-bundle -n openshift-config --from-file=ca-bundle.crt=${REGISTRY_CERT_PATH}
     ```

   - namespace にその config map が含まれていない場合は、以下のコマンドを入力します。

     ```
     ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
     export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
     export TMP_FILE=$(mktemp)

     oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' > ${TMP_FILE}
     echo >> ${TMP_FILE}
     echo \#registry.$(hostname --long) >> ${TMP_FILE}
     cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
     oc create configmap user-ca-bundle -n openshift-config --from-file=ca-bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
     ```

1.7.12.5.9. IPv4 ネットワークのホステッドクラスターをデプロイする

ホステッドクラスターは、管理クラスターにホストされたコントロールプレーンと API エンドポイントを備えた OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。

Red Hat Advanced Cluster Management のコンソールを使用してホステッドクラスターを作成できますが、次の手順ではマニフェストを使用するため、関連するアーティファクトをより柔軟に変更できます。

1.7.12.5.9.1. ホステッドクラスターオブジェクトのデプロイ

この手順では、次の値が使用されます。

- HostedCluster name: **hosted-ipv4**
- HostedCluster namespace: **clusters**
- Disconnected: **true**
- Network stack: **IPv4**

通常、HyperShift Operator は **HostedControlPlane** namespace を作成します。ただし、この場合は、HyperShift Operator が **HostedCluster** オブジェクトの調整を開始する前に、すべてのオブジェクトを含める必要があります。その後、Operator が調整プロセスを開始すると、所定の場所にあるすべてのオブジェクトを見つけることができます。

1. namespace に関する次の情報を含めて、YAML ファイルを作成します。

```
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters-hosted-ipv4
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. config map とシークレットに関する次の情報を含む YAML ファイルを作成し、**HostedCluster** デプロイメントに追加します。

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
data:
  .dockerconfigjson: xxxxxxxxx
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-ipv4-pull-secret
  namespace: clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: sshkey-cluster-hosted-ipv4
  namespace: clusters
stringData:
  id_rsa.pub: ssh-rsa xxxxxxxxx
---
apiVersion: v1
data:
  key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
kind: Secret
metadata:
  creationTimestamp: null
```

```
  name: hosted-ipv4-etcd-encryption-key
  namespace: clusters
type: Opaque
```

3. RBAC ロールを含む YAML ファイルを作成し、Assisted Service エージェントが Hosted control plane と同じ **HostedControlPlane** namespace に配置し、引き続きクラスター API で管理されるようにします。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-ipv4
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4. **HostedCluster** オブジェクトに関する情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-ipv4
  namespace: clusters
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: (1)
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
```

```
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
  platform:
    agent:
      agentNamespace: clusters-hosted-ipv4
    type: Agent
  pullSecret:
    name: hosted-ipv4-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
  secretEncryption:
    aescbc:
      activeKey:
        name: hosted-ipv4-etcd-encryption-key
    type: aescbc
  services:
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv4.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-ipv4
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

ここで、**dns.base.domain.name** は DNS ベースドメイン名であり、**4.x.y** は使用するサポートされている OpenShift Container Platform のバージョンです。

1 **imageContentSources** セクションには、ホステッドクラスター内のユーザーワークロードのミラー参照が含まれます。

5. OpenShift Container Platform リリースの HyperShift Operator リリースを指すアノテーションを **HostedCluster** オブジェクトに追加します。

   a. 次のコマンドを入力して、イメージペイロードを取得します。

   ```
   oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
   ```

   ここで、**dns.base.domain.name** は DNS ベースドメイン名であり、**4.x.y** は使用するサポートされている OpenShift Container Platform のバージョンです。

   b. 以下の出力を参照してください。

   ```
   hypershift
   sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   c. OpenShift Container Platform Images namespace を使用して、次のコマンドを入力してダイジェストを確認します。

   ```
   podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   dns.base.domain.name は DNS ベースドメイン名です。

   d. 以下の出力を参照してください。

   ```
   podman pull
   registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   Trying to pull
   registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
   Getting image source signatures
   Copying blob d8190195889e skipped: already exists
   Copying blob c71d2589fba7 skipped: already exists
   Copying blob d4dc6e74b6ce skipped: already exists
   Copying blob 97da74cc6d8f skipped: already exists
   Copying blob b70007a560c9 done
   Copying config 3a62961e6e done
   Writing manifest to image destination
   Storing signatures
   3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
   ```

   注: HostedCluster オブジェクトに設定されるリリースイメージでは、タグではなくダイジェストを使用する必要があります (例: **quay.io/openshift-release-dev/ocp-release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0**)。

6. YAML ファイルで定義したすべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに対して適用して作成します。起動するには、以下のコマンドを実行します。

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

7. Hosted control plane の出力を参照してください。

```
NAME                                          READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc                1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv              2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq                   1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d             1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht   1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8      1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf 1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf    1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts     1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7      1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4        1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr      1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp   1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j         1/1     Running   0          77s
dns-operator-6874b577f-9tc6b                  1/1     Running   0          94s
etcd-0                                        3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh   1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk              1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7        1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg             1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782           1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp               4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk      1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck               1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr              1/1     Running   0          92s
oauth-openshift-79c7bf44bf-j975g              2/2     Running   0          62s
olm-operator-767f9584c-4lcl2                  2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj          3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7   1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574     2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc   1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh                2/2     Running   0          93s
```

8. ホステッドクラスターの出力を確認します。

```
NAMESPACE   NAME         VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
clusters    hosted-ipv4            hosted-admin-kubeconfig   Partial    True        False         The hosted control plane is available
```

次に、**NodePool** オブジェクトを作成します。

1.7.12.5.9.2. ホステッドクラスターの NodePool オブジェクトの作成

NodePool は、ホステッドクラスターに関連付けられたスケーラブルなワーカーノードのセットです。**NodePool** マシンアーキテクチャーは特定のプール内で一貫性を保ち、コントロールプレーンのマシンアーキテクチャーから独立しています。

1. **NodePool** オブジェクトに関する次の情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-ipv4
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-ipv4
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64 (3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) ノードが削除された場合、ノードは再作成されないため、**autoRepair** フィールドは **false** に設定されます。

(2) **upgradeType** は **InPlace** に設定されます。これは、アップグレード中に同じベアメタルノードが再利用されることを示します。

(3) この **NodePool** に含まれるすべてのノードは、OpenShift Container Platform バージョン **4.x.y-x86_64** に基づいています。**dns.base.domain.name** を DNS ベースドメイン名に置き換え、**4.x.y** を使用したいサポートされている OpenShift Container Platform バージョンに置き換えます。

(4) **replicas** の値は **0** に設定されているため、必要に応じてスケールを変更できます。すべての手順が完了するまで、**NodePool** レプリカを 0 に保つことが重要です。

2. 次のコマンドを入力して、**NodePool** オブジェクトを作成します。

```
oc apply -f 02-nodepool.yaml
```

3. 出力を参照してください。

```
NAMESPACE   NAME        CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-ipv4   hosted    0                         False       False       4.x.y-x86_64
```

4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

次に、**InfraEnv** リソースを作成します。

1.7.12.5.9.3. ホステッドクラスターの InfraEnv リソースの作成

InfraEnv リソースは、**pullSecretRef** や **sshAuthorizedKey** などの重要な詳細を含む Assisted Service オブジェクトです。これらの詳細は、ホステッドクラスター用にカスタマイズされた Red Hat Enterprise Linux CoreOS (RHCOS) ブートイメージを作成するために使用されます。

1. **InfraEnv** リソースに関する次の情報を含めて YAML ファイルを作成し、必要に応じて値を置き換えます。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-ipv4
  namespace: clusters-hosted-ipv4
spec:
  pullSecretRef: 1
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52afsNk
w0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZFBfq2XRUxo
1wrRdor2z0Bbh93ytR+ZsDbbLlGngXaMa0Vbt+z74FqlcajbHTZ6zBmTpBVq5RHtDPgKITdpE1f
ongp7+ZXQNBlkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmnKLMW1VtCE6nJzEgWC
ufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmyhw5Sj6YboA8WICs4z+TrFP8
9LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1AmEy8QYNwpJQJNpSxzoKBjI73X
vPYYC/IjPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4KeMu7eS2tC+Kn4deJ7KwwUycx8n6RH
MeD8Qg9fITHCv3gmab8JKZJqN3hW1D378JuvmIX4V0= 2
```

1 **pullSecretRef** は、プルシークレットが使用される **InfraEnv** と同じ namespace 内の config map を参照します。

2 **sshAuthorizedKey** は、ブートイメージに配置される SSH 公開鍵を表します。SSH 鍵を使用すると、**core** ユーザーとしてワーカーノードにアクセスできます。

2. 次のコマンドを入力して、**InfraEnv** リソースを作成します。

```
oc apply -f 03-infraenv.yaml
```

3. 以下の出力を参照してください。

```
NAMESPACE             NAME     ISO CREATED AT
clusters-hosted-ipv4  hosted   2023-09-11T15:14:10Z
```

次に、ワーカーノードを作成します。

1.7.12.5.9.4. ホステッドクラスター用のワーカーノードの作成

ベアメタルプラットフォームで作業している場合、**BareMetalHost** の詳細が正しく設定されていることを確認するためにワーカーノードを作成することが重要です。

仮想マシンを使用している場合は、次の手順を実行して、Metal3 Operator が使用する空のワーカーノードを作成できます。これには、**kcli** を使用します。

1. ワーカーノードを作成するのが初めてではない場合は、まず以前の設定を削除する必要があります。これには、次のコマンドを入力してプランを削除します。

```
kcli delete plan hosted-ipv4
```

a. プランを削除するかどうかを確認するプロンプトが表示されたら、**y** と入力します。

b. プランが削除されたことを示すメッセージが表示されることを確認します。

2. 次のコマンドを入力して仮想マシンを作成します。

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:11\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211 -P name=hosted-ipv4-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:12\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212 -P name=hosted-ipv4-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv4 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv4\", \"mac\": \"aa:aa:aa:aa:02:13\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213 -P name=hosted-ipv4-worker2
```

```
systemctl restart ksushy
```

ここでは、以下のようになります。

- **start=False** は、仮想マシン (VM) が作成時に自動的に起動しないことを意味します。
- **uefi_legacy=true** は、以前の UEFI 実装との互換性を確保するために UEFI レガシーブートを使用することを意味します。
- **plan=hosted-dual** は、マシンのグループをクラスターとして識別するプラン名を示します。
- **memory=8192** および **numcpus=16** は、RAM や CPU などの仮想マシンのリソースを指定するパラメーターです。
- **discs=200,200** は、VM 内に 2 つのシンプロビジョニングディスクを作成していることを示します。
- **nets=[{"name": "dual", "mac": "aa:aa:aa:aa:02:13"}]** は、接続するネットワーク名やプライマリーインターフェイスの MAC アドレスなど、ネットワークの詳細です。
- **restart ksushy** は、**ksushy** ツールを再起動して、追加した VM をツールが確実に検出できるようにします。

3. 結果の出力を確認します。

```
+--------------------+--------+------------------+------------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                  |    Plan    | Profile
|
+--------------------+--------+------------------+------------------------------------------------+-----
--------+---------+
|   hosted-worker0   |  down  |                  |                                                 | hosted-ipv4 |
kvirt  |
```

```
|   hosted-worker1   | down |              |                                      | hosted-ipv4 |
kvirt  |
|   hosted-worker2   | down |              |                                      | hosted-ipv4 |
kvirt  |
+--------------------+------+--------------+--------------------------------------+-----
-------+---------+
```

次に、ホステッドクラスターのベアメタルホストを作成します。

1.7.12.5.9.5. ホステッドクラスターのベアメタルホスト作成

ベアメタルホスト は、物理的な詳細と論理詳細を含む **openshift-machine-api** オブジェクトで、Metal3 Operator によって識別できるようになっています。これらの詳細は、**agents** と呼ばれる他の Assisted Service オブジェクトに関連付けられています。

重要: ベアメタルホストと移行先ノードを作成する前に、仮想マシンを作成する必要があります。

ベアメタルホストを作成するには、以下の手順を実行します。

1. 次の情報を使用して YAML ファイルを作成します。
   注記: ベアメタルホストの認証情報を保持するシークレットが 1 つ以上あるため、ワーカーノードごとに少なくとも 2 つのオブジェクトを作成する必要があります。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-ipv4-worker0-bmc-secret
  namespace: clusters-hosted-ipv4
data:
  password: YWRtaW4=
  username: YWRtaW4=
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-ipv4-worker0
  namespace: clusters-hosted-ipv4
  labels:
    infraenvs.agent-install.openshift.io: hosted-ipv4 (1)
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-ipv4-worker0 (2)
spec:
  automatedCleaningMode: disabled (3)
  bmc:
    disableCertificateVerification: true (4)
    address: redfish-virtualmedia://[192.168.125.1]:9000/redfish/v1/Systems/local/hosted-ipv4-
worker0 (5)
    credentialsName: hosted-ipv4-worker0-bmc-secret (6)
  bootMACAddress: aa:aa:aa:aa:02:11 (7)
  online: true (8)
```

1. **infraenvs.agent-install.openshift.io** は、Assisted Installer オブジェクトと **BareMetalHost** オブジェクト間のリンクとして機能します。
2. **bmac.agent-install.openshift.io/hostname** は、デプロイメント中に採用されるノード名を表します。
3. **automatedCleaningMode** は、ノードが Metal3 Operator によって消去されるのを防ぎます。
4. **disableCertificateVerification** は **true** に設定され、クライアントから証明書の検証がバイパスされます。
5. **address** は、ワーカーノードのベースボード管理コントローラー (BMC) アドレスを示します。
6. **credentialsName** は、ユーザーとパスワードの認証情報が保存されるシークレットを指します。
7. **bootMACAddress** は、ノードの起動元のインターフェイス MACAddress を示します。
8. **online** は、**BareMetalHost** オブジェクトが作成された後のノードの状態を定義します。

2. 次のコマンドを入力して、**BareMetalHost** オブジェクトをデプロイします。

```
oc apply -f 04-bmh.yaml
```

プロセス中に、次の出力が確認できます。

- この出力は、プロセスがノードに到達しようとしていることを示しています。

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   registering             true         2s
clusters-hosted  hosted-worker1   registering             true         2s
clusters-hosted  hosted-worker2   registering             true         2s
```

- この出力は、ノードが起動していることを示しています。

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   provisioning            true         16s
clusters-hosted  hosted-worker1   provisioning            true         16s
clusters-hosted  hosted-worker2   provisioning            true         16s
```

- この出力は、ノードが正常に起動したことを示しています。

```
NAMESPACE        NAME             STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0   provisioned             true         67s
clusters-hosted  hosted-worker1   provisioned             true         67s
clusters-hosted  hosted-worker2   provisioned             true         67s
```

3. ノードが起動したら、次の例に示すように、namespace のエージェントに注目してください。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411             true       auto-assign
```

```
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412          true      auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413          true      auto-assign
```

エージェントは、インストールに使用できるノードを表します。ホステッドクラスターにノードを割り当てるには、ノードプールをスケールアップします。

1.7.12.5.9.6. ノードプールのスケールアップ

ベアメタルホストを作成すると、そのステータスが **Registering Provisioning**、**Provisioned** に変わります。ノードは、エージェントの **LiveISO** と、**agent** という名前のデフォルトの Pod で始まります。このエージェントは、Assisted Service Operator から OpenShift Container Platform ペイロードをインストールする指示を受け取ります。

1. ノードプールをスケールアップするには、次のコマンドを入力します。

```
oc -n clusters scale nodepool hosted-ipv4 --replicas 3
```

2. スケーリングプロセスが完了すると、エージェントがホステッドクラスターに割り当てられていることがわかります。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413   hosted    true       auto-assign
```

3. また、ノードプールレプリカが設定されていることにも注意してください。

```
NAMESPACE   NAME     CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted   hosted    3                       False        False        4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

4. ノードがクラスターに参加するまで待ちます。プロセス中に、エージェントはステージとステータスに関する最新情報を提供します。

次に、ホステッドクラスターのデプロイメントを監視します。

1.7.12.5.10. IPv4 ネットワーク用のホステッドクラスターのデプロイメントの完了

ホステッドクラスターのデプロイメントは、コントロールプレーンとデータプレーンの 2 つの観点から監視できます。

1.7.12.5.10.1. コントロールプレーンの監視

ホステッドクラスターのデプロイメント中に、次のコマンドを入力してコントロールプレーンを監視できます。

```
export KUBECONFIG=/root/.kcli/clusters/hub-ipv4/auth/kubeconfig
```

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n clusters-hosted-ipv4;echo;echo;oc get bmh -A;echo;echo;oc get agent -A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -A;echo;echo;oc get nodepool -A;echo;echo;"
```

これらのコマンドは、次のアーティファクトに関する情報を提供します。

- HyperShift Operator
- **HostedControlPlane** Pod
- ベアメタルホスト
- エージェント
- **InfraEnv** リソース
- **HostedCluster** および **NodePool** リソース

1.7.12.5.10.2. データプレーンの監視

デプロイメントプロセス中に Operator がどのように進行しているかを監視するには、次のコマンドを入力します。

```
oc get secret -n clusters-hosted-ipv4 admin-kubeconfig -o jsonpath='{.data.kubeconfig}' |base64 -d > /root/hc_admin_kubeconfig.yaml
```

```
export KUBECONFIG=/root/hc_admin_kubeconfig.yaml
```

```
watch "oc get clusterversion,nodes,co"
```

これらのコマンドは、次のアーティファクトに関する情報を提供します。

- クラスターのバージョン
- ノード (特にノードがクラスターに参加したかどうかについて)
- クラスター Operator

1.7.12.6. IPv6 ネットワーク上での Hosted control plane の設定

IPv6 ネットワーク設定は、現在 disconnected として指定されます。この指定の主な理由は、リモートレジストリーが IPv6 では機能しないためです。

IPv6 ネットワーク上で Hosted Control Plane を設定するには、次の手順を確認してください。

1. IPv6 ネットワーク用のハイパーバイザーを設定する
2. IPv6 ネットワークの DNS を設定する
3. IPv6 ネットワーク用のレジストリーをデプロイする
4. IPv6 ネットワークの管理クラスターを設定する
5. IPv6 ネットワーク用の Web サーバーを設定する

6. IPv6 ネットワークのイメージミラーリングを設定する
7. IPv6 ネットワーク用の multicluster engine Operator をデプロイする
8. IPv6 ネットワークの TLS 証明書を設定する
9. IPv6 ネットワークのホステッドクラスターをデプロイする
10. IPv6 ネットワークのデプロイメントを終了する

1.7.12.6.1. IPv6 ネットワーク用のハイパーバイザーを設定する

以下の情報は、仮想マシン環境にのみ適用されます。

1.7.12.6.1.1. 仮想 OpenShift Container Platform クラスターのパッケージへのアクセスおよびデプロイ

1. 仮想 OpenShift Container Platform 管理クラスターをデプロイするには、以下のコマンドを入力して必要なパッケージにアクセスします。

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 次のコマンドを入力して、Podman サービスを有効にして起動します。

```
systemctl enable --now podman
```

3. **kcli** を使用して OpenShift Container Platform 管理クラスターおよびその他の仮想コンポーネントをデプロイするには、以下のコマンドを入力してハイパーバイザーをインストールおよび設定します。

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd
```

```
sudo dnf -y copr enable karmab/kcli
```

```
sudo dnf -y install kcli
```

```
sudo kcli create pool -p /var/lib/libvirt/images default
```

```
kcli create host kvm -H 127.0.0.1 local
```

```
sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images
```

```
kcli create network  -c 192.168.122.0/24 default
```

1.7.12.6.1.2. ネットワークマネージャーディスパッチャーの有効化

1. ネットワークマネージャーのディスパッチャーを有効にして、仮想マシンが必要なドメイン、ルート、およびレジストリーを解決できるようにします。ネットワークマネージャーディスパッチャーを有効にするには、**/etc/NetworkManager/dispatcher.d/** ディレクトリーに次の内容を含む **Forcens** という名前のスクリプトを作成し、環境に合わせて必要に応じて値を置き換えます。

```
#!/bin/bash

export IP="2620:52:0:1306::1" 1
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE 2
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

1. OpenShift Container Platform 管理クラスターをホストするハイパーバイザーインターフェイスの IP アドレスを指すように **IP** 変数を変更します。
2. **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

2. ファイルを作成したら、次のコマンドを入力してパーミッションを追加します。

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. スクリプトを実行し、出力が **ok** を返すことを確認します。

1.7.12.6.1.3. BMC アクセスの設定

1. 仮想マシンのベースボード管理コントローラー (BMC) をシミュレートするように **ksushy** を設定します。次のコマンドを入力します。

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
```

```
kcli create sushy-service --ssl --ipv6 --port 9000
```

```
sudo systemctl daemon-reload
```

```
systemctl enable --now ksushy
```

2. 次のコマンドを入力して、サービスが正しく機能しているかどうかをテストします。

```
systemctl status ksushy
```

1.7.12.6.1.4. 接続を許可するためのハイパーバイザーシステムの設定

開発環境で作業している場合は、環境内の各種仮想ネットワークを介したさまざまなタイプの接続を許可するようにハイパーバイザーシステムを設定します。

注: 実稼働環境で作業している場合は、安全な環境を維持するために、**firewalld** サービスの適切なルールを確立し、SELinux ポリシーを設定する必要があります。

- SELinux の場合は、次のコマンドを入力します。

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- **firewalld** の場合は、次のコマンドを入力します。

```
systemctl disable --now firewalld
```

- **libvirtd** の場合は、以下のコマンドを入力します。

```
systemctl restart libvirtd
```

```
systemctl enable --now libvirtd
```

次に、環境に合わせて DNS を設定します。

1.7.12.6.1.5. 関連情報

- **kcli**, の詳細は、公式の kcli ドキュメント を参照してください。

1.7.12.6.2. IPv6 ネットワークの DNS を設定する

この手順は、仮想環境とベアメタル環境の両方で、オフライン環境とオンライン環境の両方で必須です。仮想環境とベアメタル環境の主な違いは、リソースを設定する場所にあります。ベアメタル環境では、**dnsmasq** のような軽量のソリューションではなく、Bind のようなソリューションを使用してください。

- 仮想環境で IPv6 ネットワークの DNS を設定するには、デフォルトの Ingress と DNS の動作 を参照してください。
- ベアメタル上で IPv6 ネットワークの DNS を設定するには、ベアメタル上での DNS の設定 を参照してください。

次にレジストリーをデプロイします。

1.7.12.6.3. IPv6 ネットワーク用のレジストリーをデプロイする

開発環境の場合は、Podman コンテナーを使用して、小規模な自己ホスト型レジストリーをデプロイします。実稼働環境では、Red Hat Quay、Nexus、Artifactory などのエンタープライズでホストされるレジストリーを使用します。

Podman を使用して小規模なレジストリーをデプロイするには、以下の手順を実行します。

1. 特権ユーザーとして **${HOME}** ディレクトリーにアクセスし、次のスクリプトを作成します。

```
#!/usr/bin/env bash
```

```
set -euo pipefail

PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" (1)

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi

dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -x509 -
days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER $REGISTRY_PASSWORD
```

```
podman create --name registry --net host --security-opt label=disable --replace -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e "REGISTRY_AUTH=htpasswd"
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -e
"REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1 **PULL_SECRET** の場所は、設定に適した場所に置き換えます。

2. スクリプトファイル **registry.sh** という名前を指定して保存します。スクリプトを実行すると、以下の情報がプルされます。
   - ハイパーバイザーのホスト名に基づくレジストリー名
   - 必要な認証情報およびユーザーアクセスの詳細
3. 次のように実行フラグを追加して、パーミッションを調整します。

```
chmod u+x ${HOME}/registry.sh
```

4. パラメーターを指定せずにスクリプトを実行するには、以下のコマンドを入力します。

```
${HOME}/registry.sh
```

   このスクリプトはサーバーを起動します。

5. このスクリプトは、管理目的で **systemd** サービスを使用します。スクリプトを管理する必要がある場合は、以下のコマンドを使用できます。

```
systemctl status
```

```
systemctl start
```

```
systemctl stop
```

レジストリーのルートフォルダーは **/opt/registry** ディレクトリー内にあり、次のサブディレクトリーが含まれています。

- **certs** には TLS 証明書が含まれます。
- **auth** には認証情報が含まれます。
- **data** にはレジストリーイメージが含まれます。
- **conf** にはレジストリー設定が含まれています。

1.7.12.6.4. IPv6 ネットワークの管理クラスターの設定

OpenShift Container Platform 管理クラスターを設定するには、oc-mirror を使用できます。または

OpenShift Container Platform 管理クラスターを設定するには、dev-scripts を使用できます。または、仮想マシンをベースにしている場合は、**kcli** ツールを使用できます。以下は、**kcli** ツールに固有のものです。

1. ハイパーバイザーで使用するために適切なネットワークの準備が完了していることを確認します。ネットワークは、管理クラスターとホステッドクラスターの両方をホストします。以下の **kcli** コマンドを入力します。

```
kcli create network -c 2620:52:0:1305::0/64 -P dhcp=false -P dns=false --domain
dns.base.domain.name --nodhcp ipv6
```

   ここでは、以下のようになります。

   - **-c** は、ネットワークの CIDR を指定します。
   - **-p dhcp=false** は、設定した **dnsmasq** によって処理される DHCP を無効にするようにネットワークを設定します。
   - **-P dns=false は、** DNS を無効にするようにネットワークを設定します。これも、設定した **dnsmasq** によって処理されます。
   - **--domain** は、検索するドメインを設定します。
   - **dns.base.domain.name** は DNS ベースドメイン名です。
   - **ipv6** は、作成するネットワークの名前です。

2. ネットワークを作成したら、以下の出力を確認します。

```
[root@hypershiftbm ~]# kcli list network
Listing Networks...
+---------+--------+--------------------+-------+------------------+------+
| Network |  Type  |        Cidr        |  Dhcp |      Domain      | Mode |
+---------+--------+--------------------+-------+------------------+------+
| default | routed |   192.168.122.0/24  |  True |     default      | nat  |
| ipv4    | routed |   192.168.125.0/24  | False | dns.base.domain.name | nat  |
| ipv4    | routed | 2620:52:0:1305::/64 | False | dns.base.domain.name | nat  |
+---------+--------+--------------------+-------+------------------+------+
```

```
[root@hypershiftbm ~]# kcli info network ipv6
Providing information about network ipv6...
cidr: 2620:52:0:1305::/64
dhcp: false
domain: dns.base.domain.name
mode: nat
plan: kvirt
type: routed
```

3. OpenShift Container Platform 管理クラスターをデプロイできるように、プルシークレットと **kcli** プランファイルが配置されていることを確認します。

   a. プルシークレットが **kcli** プランと同じフォルダーにあり、プルシークレットファイルの名前が **openshift_pull.json** であることを確認します。

   b. OpenShift Container Platform 定義を含む **kcli** プランを **mgmt-compact-hub-ipv6.yaml** ファイルに追加します。ご使用の環境に合わせてファイルの内容を更新してください。

```
plan: hub-ipv6
force: true
version: nightly
tag: "4.x.y-x86_64"
cluster: "hub-ipv6"
ipv6: true
domain: dns.base.domain.name
api_ip: 2620:52:0:1305::2
ingress_ip: 2620:52:0:1305::3
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: false
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: ipv6
users_dev: developer
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: ipv6-virtual-network
metallb_ranges:
- 2620:52:0:1305::150-2620:52:0:1305::190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:10
- hub-ctlplane-0:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:01
- hub-ctlplane-1:
    nets:
```

```
    - name: ipv6
      mac: aa:aa:aa:aa:03:02
- hub-ctlplane-2:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:03:03
```

4. 管理クラスターをプロビジョニングするには、以下のコマンドを入力します。

```
kcli create cluster openshift --pf mgmt-compact-hub-ipv6.yaml
```

1.7.12.6.4.1. 関連情報

- **kcli** プランファイルのパラメーターの詳細は、**kcli** 公式ドキュメントの parameters.yml の作成 を参照してください。

1.7.12.6.5. IPv6 ネットワーク用の Web サーバーを設定する

ホステッドクラスターとしてデプロイする OpenShift Container Platform リリースに関連付けられた Red Hat Enterprise Linux CoreOS (RHCOS) イメージをホストするには、追加の Web サーバーを設定する必要があります。

Web サーバーを設定するには、以下の手順を実行します。

1. 以下のコマンドを入力して、使用する OpenShift Container Platform リリースから **openshift-install** バイナリーを展開します。

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}"
```

2. 次のスクリプトを実行します。このスクリプトは、**/opt/srv** ディレクトリーにフォルダーを作成します。このフォルダーには、ワーカーノードをプロビジョニングするための RHCOS イメージが含まれています。

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*

## Run Webserver
podman ps --noheading | grep -q websrv-ai
if [[ $? == 0 ]];then
    echo "Launching Registry pod..."
```

```
    /usr/bin/podman run --name websrv-ai --net host -v /opt/srv:/usr/local/apache2/htdocs:z
quay.io/alosadag/httpd:p8080
fi
```

1. **ROOTFS_IMG_URL** 値は OpenShift CI Release ページにあります。
2. **LIVE_ISO_URL** 値は、OpenShift CI リリースページで確認できます。

ダウンロードが完了すると、コンテナーが実行され、Web サーバー上でイメージをホストします。このコンテナーは公式 HTTPd イメージのバリエーションを使用しているので、IPv6 ネットワークでの動作も可能になります。

1.7.12.6.6. IPv6 ネットワークのイメージミラーリングを設定する

イメージミラーリングは、**registry.redhat.com** や **quay.io** などの外部レジストリーからイメージを取得し、プライベートレジストリーに保存するプロセスです。

1.7.12.6.6.1. ミラーリングプロセスの完了

注: ミラーリングプロセスは、レジストリーサーバーの実行後に開始してください。

次の手順では、**ImageSetConfiguration** オブジェクトを使用するバイナリーである、**oc-mirror** ツールが使用されます。このファイルで、以下の情報を指定できます。

- ミラーリングする OpenShift Container Platform バージョン。バージョンは **quay.io** にあります。
- ミラーリングする追加の Operator。パッケージは個別に選択します。
- リポジトリーに追加する追加のイメージ。

イメージのミラーリングを設定するには、以下の手順を実行します。

1. **${HOME}/.docker/config.json** ファイルが、ミラーリング元のレジストリーとイメージのプッシュ先のプライベートレジストリーで更新されていることを確認します。
2. 次の例を使用して、ミラーリングに使用する **ImageSetConfiguration** オブジェクトを作成します。環境に合わせて必要に応じて値を置き換えます。

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest 1
mirror:
  platform:
    channels:
    - name: candidate-4.14
      minVersion: 4.x.y-x86_64
      maxVersion: 4.x.y-x86_64
      type: ocp
    graph: true
  additionalImages:
  - name: quay.io/karmab/origin-keepalived-ipfailover:latest
  - name: quay.io/karmab/kubectl:latest
```

```
    - name: quay.io/karmab/haproxy:latest
    - name: quay.io/karmab/mdns-publisher:latest
    - name: quay.io/karmab/origin-coredns:latest
    - name: quay.io/karmab/curl:latest
    - name: quay.io/karmab/kcli:latest
    - name: quay.io/user-name/trbsht:latest
    - name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
    - name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
    packages:
    - name: lvms-operator
    - name: local-storage-operator
    - name: odf-csi-addons-operator
    - name: odf-operator
    - name: mcg-operator
    - name: ocs-operator
    - name: metallb-operator
```

1 **dns.base.domain.name** を DNS ベースドメイン名に置き換え、**4.x.y** を使用したいサポートされている OpenShift Container Platform バージョンに置き換えます。

3. 次のコマンドを入力して、ミラーリングプロセスを開始します。

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

ミラーリングプロセスが完了すると、**oc-mirror-workspace/results-XXXXXX/** という名前の新しいフォルダーが作成されます。このフォルダーには、ICSP と、ホステッドクラスターに適用するカタログソースが含まれます。

4. **oc adm release Mirror** コマンドを使用して、OpenShift Container Platform の夜間バージョンまたは CI バージョンをミラーリングします。以下のコマンドを入力します。

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
  --from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
  --to=${REGISTRY}/openshift/release \
  --to-release-image=${REGISTRY}/openshift/release-images:4.x.y-x86_64
```

4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

5. 非接続ネットワークへのインストール の手順に従って、最新の multicluster engine Operator イメージをミラーリングします。

1.7.12.6.6.2. 管理クラスターでのオブジェクトの適用

ミラーリングプロセスが完了したら、管理クラスターに 2 つのオブジェクトを適用する必要があります。

- イメージコンテンツソースポリシー (ICSP) またはイメージダイジェストミラーセット (IDMS)
- カタログソース

oc-mirror ツールを使用すると、出力アーティファクトは **oc-mirror-workspace/results-XXXXXX/** という名前のフォルダーに保存されます。

ICSP または IDMS は、ノードを再起動せずに、各ノードで kubelet を再起動する **MachineConfig** 変更を開始します。ノードが **READY** としてマークされたら、新しく生成されたカタログソースを適用する必要があります。

カタログソースは、カタログイメージのダウンロードや処理を行い、そのイメージに含まれるすべての **packagemanifests** を取得するなど、**openshift-marketplace** Operator でアクションを開始します。

1. 新しいソースを確認するには、新しい **CatalogSource** をソースとして使用して次のコマンドを実行します。

```
oc get packagemanifest
```

2. アーティファクトを適用するには、次の手順を実行します。
   a. 次のコマンドを入力して、ICSP または IDMS アーティファクトを作成します。

```
oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
```

   b. ノードの準備が完了するまで待ってから、次のコマンドを入力します。

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

1.7.12.6.6.3. 関連情報

- 仮想環境で作業している場合は、ミラーリングを設定した後、OpenShift Virtualization 上のホストされたコントロールプレーンの前提条件 を満たしていることを確認してください。
- OpenShift Container Platform の夜間ミラーリングまたは CI バージョンのミラーリングの詳細は、oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング を参照してください。

1.7.12.6.7. IPv6 ネットワーク用の multicluster engine Operator をデプロイする

multicluster engine Operator は、プロバイダー間でクラスターをデプロイメントする場合に重要な役割を果たします。Red Hat Advanced Cluster Management をすでにインストールしている場合は、multicluster engine Operator は自動的にインストールされるため、インストールする必要はありません。

multicluster engine Operator がインストールされていない場合は、次のドキュメントを参照して、前提条件とインストール手順を確認してください。

- multicluster engine operator を使用したクラスターライフサイクルについて
- multicluster engine operator のインストールとアップグレード

1.7.12.6.7.1. AgentServiceConfig リソースのデプロイ

AgentServiceConfig カスタムリソースは、multicluster engine operator の一部である Assisted Service アドオンの重要なコンポーネントです。このコンポーネントは、ベアメタルクラスターをデプロイメントします。アドオンが有効な場合に、**AgentServiceConfig** リソースをデプロイしてアドオンを設定します。

AgentServiceConfig リソースの設定に加えて、multicluster engine Operator が非接続環境で適切に機能するように、追加の config map を含める必要があります。

1. 次の config map を追加してカスタムレジストリーを設定します。これには、デプロイメントをカスタマイズするための非接続環境の情報が含まれています。

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" 1

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

1 **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

オブジェクトには、以下の 2 つのフィールドが含まれます。

- カスタム CA: このフィールドには、デプロイメントのさまざまなプロセスに読み込まれる認証局 (CA) が含まれます。
- レジストリー: **Registries.conf** フィールドには、元のソースレジストリーではなくミラーレジストリーから使用する必要があるイメージと namespace に関する情報が含まれています。

2. 次の例に示すように、**AssistedServiceConfig** オブジェクトを追加して、Assisted Service を設定します。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  annotations:
```

```
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-service-config
1
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries 2
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
  osImages: 3
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img 4
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-0-
live.x86_64.iso
    version: 414.92.202308281054-0
```

1. **metadata.annotations"unsupported.agent-install.openshift.io/assisted-service-configmap"** アノテーションは、Operator が動作をカスタマイズするために使用する config map 名を参照します。
2. **spec.mirrorRegistryRef.name** アノテーションは、Assisted Service Operator が使用する非接続のレジストリー情報を含む config map を指します。この config map は、デプロイメントプロセス中にこれらのリソースを追加します。
3. **spec.osImages** フィールドには、この Operator がデプロイできるさまざまなバージョンが含まれています。このフィールドは必須です。この例では、**RootFS** ファイルと **LiveISO** ファイルがすでにダウンロードされていることを前提としています。
4. **rootFSUrl フィールド** と **url** フィールドで、**dns.base.domain.name** を DNS ベースドメイン名に置き換えます。

3. すべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに適用し、これらのオブジェクトをデプロイします。起動するには、以下のコマンドを実行します。

```
oc apply -f agentServiceConfig.yaml
```

このコマンドは、次の出力例に示すように、2 つの Pod をトリガーします。

```
assisted-image-service-0                  1/1     Running   2          11d 1
assisted-service-668b49548-9m7xw          2/2     Running   5          11d 2
```

1 **Assisted-image-service** Pod は、デプロイするクラスターごとにカスタマイズされた、Red Hat Enterprise Linux CoreOS (RHCOS) 起動イメージテンプレートを作成します。

2 **assisted-service** は Operator を参照します。

1.7.12.6.8. IPv6 ネットワークの TLS 証明書を設定する

オフライン環境で Hosted control plane を設定するプロセスに、いくつかの TLS 証明書が関与します。認証局 (CA) を管理クラスターに追加するには、OpenShift Container Platform コントロールプレーンおよびワーカーノード内の以下のファイルの内容を変更する必要があります。

- **/etc/pki/ca-trust/extracted/pem/**
- **/etc/pki/ca-trust/source/anchors**
- **/etc/pki/tls/certs/**

CA を管理クラスターに追加するには、次の手順を実行します。

1. OpenShift Container Platform の公式ドキュメントの CA バンドルの更新 の手順を完了します。この方法には、CA を OpenShift Container Platform ノードにデプロイする **image-registry-operator** の使用が含まれます。
2. この方法が実際の状況に該当しない場合は、管理クラスター内の **openshift-config** namespace に **user-ca-bundle** という名前の config map が含まれているかどうかを確認してください。
   - namespace にその config map が含まれている場合は、次のコマンドを入力します。

     ```
     ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
     export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

     oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
     bundle.crt=${REGISTRY_CERT_PATH}
     ```

   - namespace にその config map が含まれていない場合は、以下のコマンドを入力します。

     ```
     ## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
     export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
     export TMP_FILE=$(mktemp)

     oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' >
     ${TMP_FILE}
     echo >> ${TMP_FILE}
     echo \#registry.$(hostname --long) >> ${TMP_FILE}
     cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
     oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
     bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
     ```

1.7.12.6.9. IPv6 ネットワークのホステッドクラスターをデプロイする

ホステッドクラスターは、管理クラスターにホストされたコントロールプレーンと API エンドポイントを備えた OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。

Red Hat Advanced Cluster Management のコンソールを使用してホステッドクラスターを作成できますが、次の手順ではマニフェストを使用するため、関連するアーティファクトをより柔軟に変更できます。

1.7.12.6.9.1. ホステッドクラスターオブジェクトのデプロイ

この手順では、次の値が使用されます。

- HostedCluster name: **hosted-ipv6**
- HostedCluster namespace: **clusters**
- Disconnected: **true**
- Network stack: **IPv6**

通常、HyperShift Operator は **HostedControlPlane** namespace を作成します。ただし、この場合は、HyperShift Operator が **HostedCluster** オブジェクトの調整を開始する前に、すべてのオブジェクトを含める必要があります。その後、Operator が調整プロセスを開始すると、所定の場所にあるすべてのオブジェクトを見つけることができます。

1. namespace に関する次の情報を含めて、YAML ファイルを作成します。

```
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters-hosted-ipv6
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. config map とシークレットに関する次の情報を含む YAML ファイルを作成し、**HostedCluster** デプロイメントに追加します。

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
data:
```

```
  .dockerconfigjson: xxxxxxxxx
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-ipv6-pull-secret
  namespace: clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: sshkey-cluster-hosted-ipv6
  namespace: clusters
stringData:
  id_rsa.pub: ssh-rsa xxxxxxxxx
---
apiVersion: v1
data:
  key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-ipv6-etcd-encryption-key
  namespace: clusters
type: Opaque
```

3. RBAC ロールを含む YAML ファイルを作成し、Assisted Service エージェントが Hosted control plane と同じ **HostedControlPlane** namespace に配置し、引き続きクラスター API で管理されるようにします。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-ipv6
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4. **HostedCluster** オブジェクトに関する情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-ipv6
  namespace: clusters
  annotations:
    hypershift.openshift.io/control-plane-operator-image: registry.ocp-edge-cluster-
0.qe.lab.redhat.com:5005/openshift/release@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f
7a73801b2c06c639bf664876617f8
```

```
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: 1
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
  platform:
    agent:
      agentNamespace: clusters-hosted-ipv6
    type: Agent
  pullSecret:
    name: hosted-ipv6-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
  secretEncryption:
    aescbc:
      activeKey:
        name: hosted-ipv6-etcd-encryption-key
    type: aescbc
  services:
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
```

```
    type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-ipv6.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-ipv6
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

ここで、**dns.base.domain.name** は DNS ベースドメイン名であり、**4.x.y** は使用するサポートされている OpenShift Container Platform のバージョンです。

1 **imageContentSources** セクションには、ホステッドクラスター内のユーザーワークロードのミラー参照が含まれます。

5. OpenShift Container Platform リリースの HyperShift Operator リリースを指すアノテーションを **HostedCluster** オブジェクトに追加します。

   a. 次のコマンドを入力して、イメージペイロードを取得します。

   ```
   oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
   ```

   ここで、**dns.base.domain.name** は DNS ベースドメイン名であり、**4.x.y** は使用するサポートされている OpenShift Container Platform のバージョンです。

   b. 以下の出力を参照してください。

   ```
   hypershift
   sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   c. OpenShift Container Platform Images namespace を使用して、次のコマンドを入力してダイジェストを確認します。

   ```
   podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   dns.base.domain.name は DNS ベースドメイン名です。

   d. 以下の出力を参照してください。

```
podman pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8
Trying to pull
registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
Getting image source signatures
Copying blob d8190195889e skipped: already exists
Copying blob c71d2589fba7 skipped: already exists
Copying blob d4dc6e74b6ce skipped: already exists
Copying blob 97da74cc6d8f skipped: already exists
Copying blob b70007a560c9 done
Copying config 3a62961e6e done
Writing manifest to image destination
Storing signatures
3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
```

注: HostedCluster オブジェクトに設定されるリリースイメージでは、タグではなくダイジェストを使用する必要があります (例: **quay.io/openshift-release-dev/ocp-release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0**)。

6. YAML ファイルで定義したすべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに対して適用して作成します。起動するには、以下のコマンドを実行します。

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

7. Hosted control plane の出力を参照してください。

```
NAME                                          READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc                1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv              2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq                   1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d             1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht   1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8      1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf 1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf    1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts     1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7      1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4        1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr      1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp   1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j         1/1     Running   0          77s
dns-operator-6874b577f-9tc6b                  1/1     Running   0          94s
etcd-0                                        3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh   1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk              1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7        1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg             1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782           1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp               4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk      1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck               1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr              1/1     Running   0          92s
```

```
oauth-openshift-79c7bf44bf-j975g                  2/2     Running   0          62s
olm-operator-767f9584c-4lcl2                      2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj              3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7     1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574         2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc   1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh                    2/2     Running   0          93s
```

8. ホステッドクラスターの出力を確認します。

```
NAMESPACE   NAME        VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
clusters    hosted-ipv6             hosted-admin-kubeconfig   Partial    True        False         The hosted control plane is available
```

次に、**NodePool** オブジェクトを作成します。

1.7.12.6.9.2. ホステッドクラスターの NodePool オブジェクトの作成

NodePool は、ホステッドクラスターに関連付けられたスケーラブルなワーカーノードのセットです。**NodePool** マシンアーキテクチャーは特定のプール内で一貫性を保ち、コントロールプレーンのマシンアーキテクチャーから独立しています。

1. **NodePool** オブジェクトに関する次の情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-ipv6
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-ipv6
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64 (3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) ノードが削除された場合、ノードは再作成されないため、**autoRepair** フィールドは **false** に設定されます。

(2) **upgradeType** は **InPlace** に設定されます。これは、アップグレード中に同じベアメタルノードが再利用されることを示します。

(3)

この **NodePool** に含まれるすべてのノードは、OpenShift Container Platform バージョン **4.x.y-x86_64** に基づいています。**dns.base.domain.name** を DNS ベースドメイン名に置

4 **replicas** の値は **0** に設定されているため、必要に応じてスケールを変更できます。すべての手順が完了するまで、**NodePool** レプリカを 0 に保つことが重要です。

2. 次のコマンドを入力して、**NodePool** オブジェクトを作成します。

```
oc apply -f 02-nodepool.yaml
```

3. 出力を参照してください。

```
NAMESPACE   NAME         CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-ipv6   hosted    0                 False       False       4.x.y-x86_64
```

次に、**InfraEnv** リソースを作成します。

1.7.12.6.9.3. ホステッドクラスターの InfraEnv リソースの作成

InfraEnv リソースは、**pullSecretRef** や **sshAuthorizedKey** などの重要な詳細を含む Assisted Service オブジェクトです。これらの詳細は、ホステッドクラスター用にカスタマイズされた Red Hat Enterprise Linux CoreOS (RHCOS) ブートイメージを作成するために使用されます。

1. **InfraEnv** リソースに関する次の情報を含めて YAML ファイルを作成し、必要に応じて値を置き換えます。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-ipv6
  namespace: clusters-hosted-ipv6
spec:
  pullSecretRef: 1
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52afsNk
w0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZFBfq2XRUxo
1wrRdor2z0Bbh93ytR+ZsDbbLlGngXaMa0Vbt+z74FqlcajbHTZ6zBmTpBVq5RHtDPgKITdpE1f
ongp7+ZXQNBlkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmnKLMW1VtCE6nJzEgWC
ufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmyhw5Sj6YboA8WICs4z+TrFP8
9LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1AmEy8QYNwpJQJNpSxzoKBjI73X
vPYYC/IjPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4KeMu7eS2tC+Kn4deJ7KwwUycx8n6RH
MeD8Qg9fITHCv3gmab8JKZJqN3hW1D378JuvmIX4V0= 2
```

1 **pullSecretRef** は、プルシークレットが使用される **InfraEnv** と同じ namespace 内の config map を参照します。

2 **sshAuthorizedKey** は、ブートイメージに配置される SSH 公開鍵を表します。SSH 鍵を使用すると、**core** ユーザーとしてワーカーノードにアクセスできます。

2. 次のコマンドを入力して、**InfraEnv** リソースを作成します。

```
oc apply -f 03-infraenv.yaml
```

3. 以下の出力を参照してください。

```
NAMESPACE              NAME     ISO CREATED AT
clusters-hosted-ipv6   hosted   2023-09-11T15:14:10Z
```

次に、ワーカーノードを作成します。

1.7.12.6.9.4. ホステッドクラスター用のワーカーノードの作成

ベアメタルプラットフォームで作業している場合、**BareMetalHost** の詳細が正しく設定されていることを確認するためにワーカーノードを作成することが重要です。

仮想マシンを使用している場合は、次の手順を実行して、Metal3 Operator が使用する空のワーカーノードを作成できます。これには、**kcli** ツールを使用します。

1. ワーカーノードを作成するのが初めてではない場合は、まず以前の設定を削除する必要があります。これには、次のコマンドを入力してプランを削除します。

```
kcli delete plan hosted-ipv6
```

   a. プランを削除するかどうかを確認するプロンプトが表示されたら、**y** と入力します。

   b. プランが削除されたことを示すメッセージが表示されることを確認します。

2. 次のコマンドを入力して仮想マシンを作成します。

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:11\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211 -P name=hosted-ipv6-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:12\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212 -P name=hosted-ipv6-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-ipv6 -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"ipv6\", \"mac\": \"aa:aa:aa:aa:02:13\"}"]
-P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213 -P name=hosted-ipv6-worker2
```

```
systemctl restart ksushy
```

   ここでは、以下のようになります。

   - **start=False** は、仮想マシン (VM) が作成時に自動的に起動しないことを意味します。
   - **uefi_legacy=true** は、以前の UEFI 実装との互換性を確保するために UEFI レガシーブートを使用することを意味します。
   - **plan=hosted-dual** は、マシンのグループをクラスターとして識別するプラン名を示します。

- **memory=8192** および **numcpus=16** は、RAM や CPU などの仮想マシンのリソースを指定するパラメーターです。
- **discs=200,200** は、VM 内に 2 つのシンプロビジョニングディスクを作成していることを示します。
- **nets=[{"name": "dual", "mac": "aa:aa:aa:aa:02:13"}]** は、接続するネットワーク名やプライマリーインターフェイスの MAC アドレスなど、ネットワークの詳細です。
- **restart ksushy** は、**ksushy** ツールを再起動して、追加した VM をツールが確実に検出できるようにします。

3. 結果の出力を確認します。

```
+--------------------+--------+------------------+------------------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                  |    Plan    | Profile
|
+--------------------+--------+------------------+------------------------------------------------------+-----
--------+---------+
|   hosted-worker0   |  down  |                  |                                          | hosted-ipv6 |
kvirt  |
|   hosted-worker1   |  down  |                  |                                          | hosted-ipv6 |
kvirt  |
|   hosted-worker2   |  down  |                  |                                          | hosted-ipv6 |
kvirt  |
+--------------------+--------+------------------+------------------------------------------------------+-----
--------+---------+
```

次に、ホステッドクラスターのベアメタルホストを作成します。

1.7.12.6.9.5. ホステッドクラスターのベアメタルホスト作成

ベアメタルホスト は、物理的な詳細と論理詳細を含む **openshift-machine-api** オブジェクトで、Metal3 Operator によって識別できるようになっています。これらの詳細は、**agents** と呼ばれる他の Assisted Service オブジェクトに関連付けられています。

重要: ベアメタルホストと移行先ノードを作成する前に、仮想マシンを作成する必要があります。

ベアメタルホストを作成するには、以下の手順を実行します。

1. 次の情報を使用して YAML ファイルを作成します。
   注記: ベアメタルホストの認証情報を保持するシークレットが 1 つ以上あるため、ワーカーノードごとに少なくとも 2 つのオブジェクトを作成する必要があります。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-ipv6-worker0-bmc-secret
  namespace: clusters-hosted-ipv6
data:
  password: YWRtaW4=
  username: YWRtaW4=
type: Opaque
---
```

```
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-ipv6-worker0
  namespace: clusters-hosted-ipv6
  labels:
    infraenvs.agent-install.openshift.io: hosted-ipv6 1
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-ipv6-worker0 2
spec:
  automatedCleaningMode: disabled 3
  bmc:
    disableCertificateVerification: true 4
    address: redfish-virtualmedia://[192.168.125.1]:9000/redfish/v1/Systems/local/hosted-ipv6-
worker0 5
    credentialsName: hosted-ipv6-worker0-bmc-secret 6
  bootMACAddress: aa:aa:aa:aa:03:11 7
  online: true 8
```

1. **infraenvs.agent-install.openshift.io** は、Assisted Installer オブジェクトと **BareMetalHost** オブジェクト間のリンクとして機能します。
2. **bmac.agent-install.openshift.io/hostname** は、デプロイメント中に採用されるノード名を表します。
3. **automatedCleaningMode** は、ノードが Metal3 Operator によって消去されるのを防ぎます。
4. **disableCertificateVerification** は **true** に設定され、クライアントから証明書の検証がバイパスされます。
5. **address** は、ワーカーノードのベースボード管理コントローラー (BMC) アドレスを示します。
6. **credentialsName** は、ユーザーとパスワードの認証情報が保存されるシークレットを指します。
7. **bootMACAddress** は、ノードの起動元のインターフェイス MAC アドレスを示します。
8. **online** は、**BareMetalHost** オブジェクトが作成された後のノードの状態を定義します。

2. 次のコマンドを入力して、**BareMetalHost** オブジェクトをデプロイします。

```
oc apply -f 04-bmh.yaml
```

プロセス中に、次の出力が確認できます。

- この出力は、プロセスがノードに到達しようとしていることを示しています。

```
NAMESPACE        NAME           STATE       CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 registering             true     2s
clusters-hosted  hosted-worker1 registering             true     2s
clusters-hosted  hosted-worker2 registering             true     2s
```

- この出力は、ノードが起動していることを示しています。

```
NAMESPACE        NAME           STATE         CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioning            true        16s
clusters-hosted  hosted-worker1 provisioning            true        16s
clusters-hosted  hosted-worker2 provisioning            true        16s
```

- この出力は、ノードが正常に起動したことを示しています。

```
NAMESPACE        NAME           STATE         CONSUMER   ONLINE   ERROR   AGE
clusters-hosted  hosted-worker0 provisioned            true        67s
clusters-hosted  hosted-worker1 provisioned            true        67s
clusters-hosted  hosted-worker2 provisioned            true        67s
```

3. ノードが起動したら、次の例に示すように、namespace のエージェントに注目してください。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411             true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412             true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413             true       auto-assign
```

エージェントは、インストールに使用できるノードを表します。ホステッドクラスターにノードを割り当てるには、ノードプールをスケールアップします。

1.7.12.6.9.6. ノードプールのスケールアップ

ベアメタルホストを作成すると、そのステータスが **Registering Provisioning**、**Provisioned** に変わります。ノードは、エージェントの **LiveISO** と、**agent** という名前のデフォルトの Pod で始まります。このエージェントは、Assisted Service Operator から OpenShift Container Platform ペイロードをインストールする指示を受け取ります。

1. ノードプールをスケールアップするには、次のコマンドを入力します。

```
oc -n clusters scale nodepool hosted-ipv6 --replicas 3
```

2. スケーリングプロセスが完了すると、エージェントがホステッドクラスターに割り当てられていることがわかります。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0211   hosted    true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0212   hosted    true       auto-assign
clusters-hosted  aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0213   hosted    true       auto-assign
```

3. また、ノードプールレプリカが設定されていることにも注意してください。

```
NAMESPACE   NAME    CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION          UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted  hosted    3                             False         False         4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

4. ノードがクラスターに参加するまで待ちます。プロセス中に、エージェントはステージとステータスに関する最新情報を提供します。

次に、ホステッドクラスターのデプロイメントを監視します。

1.7.12.6.10. IPv6 ネットワークのホステッドクラスターのデプロイメントを完了する

ホステッドクラスターのデプロイメントは、コントロールプレーンとデータプレーンの 2 つの観点から監視できます。

1.7.12.6.10.1. コントロールプレーンの監視

ホステッドクラスターのデプロイメント中に、次のコマンドを入力してコントロールプレーンを監視できます。

```
export KUBECONFIG=/root/.kcli/clusters/hub-ipv4/auth/kubeconfig
```

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n clusters-hosted-ipv4;echo;echo;oc get bmh -A;echo;echo;oc get agent -A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -A;echo;echo;oc get nodepool -A;echo;echo;"
```

これらのコマンドは、次のアーティファクトに関する情報を提供します。

- HyperShift Operator
- **HostedControlPlane** Pod
- ベアメタルホスト
- エージェント
- **InfraEnv** リソース
- **HostedCluster** および **NodePool** リソース

1.7.12.6.10.2. データプレーンの監視

デプロイメントプロセス中に Operator がどのように進行しているかを監視するには、次のコマンドを入力します。

```
oc get secret -n clusters-hosted-ipv4 admin-kubeconfig -o jsonpath='{.data.kubeconfig}' |base64 -d > /root/hc_admin_kubeconfig.yaml
```

```
export KUBECONFIG=/root/hc_admin_kubeconfig.yaml
```

```
watch "oc get clusterversion,nodes,co"
```

これらのコマンドは、次のアーティファクトに関する情報を提供します。

- クラスターのバージョン
- ノード (特にノードがクラスターに参加したかどうかについて)
- クラスター Operator

1.7.12.7. デュアルスタックネットワーク上での Hosted control plane の設定 (テクノロジープレビュー)

Hosted control plane のコンテキストでは、非接続環境は、インターネットに接続されておらず、Hosted control plane をベースとして使用する OpenShift Container Platform クラスターです。リモートレジストリーは IPv6 では機能しないため、非接続環境のデュアルスタックネットワークでのみ Hosted Control Plane を設定できます。

デュアルスタックネットワークで Hosted Control Plane を設定するには、次の手順を確認します。

1. デュアルスタックネットワーク用のハイパーバイザーの設定
2. デュアルスタックネットワーク用の DNS の設定
3. デュアルスタックネットワーク用のレジストリーのデプロイ
4. デュアルスタックネットワークの管理クラスターの設定
5. デュアルスタックネットワーク用の Web サーバーの設定
6. デュアルスタックネットワークのイメージミラーリングの設定
7. デュアルスタックネットワーク用の multicluster engine Operator のデプロイ
8. デュアルスタックネットワークの TLS 証明書の設定
9. デュアルスタックネットワーク用のホステッドクラスターのデプロイ
10. デュアルスタックネットワークのデプロイメントの終了

1.7.12.7.1. デュアルスタックネットワーク用のハイパーバイザーの設定

以下の情報は、仮想マシン環境にのみ適用されます。

1.7.12.7.1.1. 仮想 OpenShift Container Platform クラスターのパッケージへのアクセスおよびデプロイ

1. 仮想 OpenShift Container Platform 管理クラスターをデプロイするには、以下のコマンドを入力して必要なパッケージにアクセスします。

```
sudo dnf install dnsmasq radvd vim golang podman bind-utils net-tools httpd-tools tree htop strace tmux -y
```

2. 次のコマンドを入力して、Podman サービスを有効にして起動します。

```
systemctl enable --now podman
```

3. **kcli** を使用して OpenShift Container Platform 管理クラスターおよびその他の仮想コンポーネントをデプロイするには、以下のコマンドを入力してハイパーバイザーをインストールおよび設定します。

```
sudo yum -y install libvirt libvirt-daemon-driver-qemu qemu-kvm
```

```
sudo usermod -aG qemu,libvirt $(id -un)
```

```
sudo newgrp libvirt
```

```
sudo systemctl enable --now libvirtd

sudo dnf -y copr enable karmab/kcli

sudo dnf -y install kcli

sudo kcli create pool -p /var/lib/libvirt/images default

kcli create host kvm -H 127.0.0.1 local

sudo setfacl -m u:$(id -un):rwx /var/lib/libvirt/images

kcli create network  -c 192.168.122.0/24 default
```

1.7.12.7.1.2. ネットワークマネージャーディスパッチャーの有効化

1. ネットワークマネージャーのディスパッチャーを有効にして、仮想マシンが必要なドメイン、ルート、およびレジストリーを解決できるようにします。ネットワークマネージャーディスパッチャーを有効にするには、**/etc/NetworkManager/dispatcher.d/** ディレクトリーに次の内容を含む **Forcens** という名前のスクリプトを作成し、環境に合わせて必要に応じて値を置き換えます。

```
#!/bin/bash

export IP="192.168.126.1" (1)
export BASE_RESOLV_CONF="/run/NetworkManager/resolv.conf"

if ! [[ `grep -q "$IP" /etc/resolv.conf` ]]; then
export TMP_FILE=$(mktemp /etc/forcedns_resolv.conf.XXXXXX)
cp $BASE_RESOLV_CONF $TMP_FILE
chmod --reference=$BASE_RESOLV_CONF $TMP_FILE
sed -i -e "s/dns.base.domain.name//" -e "s/search /& dns.base.domain.name /" -e
"0,/nameserver/s/nameserver/& $IP\n&/" $TMP_FILE (2)
mv $TMP_FILE /etc/resolv.conf
fi
echo "ok"
```

(1) OpenShift Container Platform 管理クラスターをホストするハイパーバイザーインターフェイスの IP アドレスを指すように **IP** 変数を変更します。

(2) **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

2. ファイルを作成したら、次のコマンドを入力してパーミッションを追加します。

```
chmod 755 /etc/NetworkManager/dispatcher.d/forcedns
```

3. スクリプトを実行し、出力が **ok** を返すことを確認します。

1.7.12.7.1.3. BMC アクセスの設定

1. 仮想マシンのベースボード管理コントローラー (BMC) をシミュレートするように **ksushy** を設定します。次のコマンドを入力します。

```
sudo dnf install python3-pyOpenSSL.noarch python3-cherrypy -y
```

```
kcli create sushy-service --ssl --ipv6 --port 9000
```

```
sudo systemctl daemon-reload
```

```
systemctl enable --now ksushy
```

2. 次のコマンドを入力して、サービスが正しく機能しているかどうかをテストします。

```
systemctl status ksushy
```

1.7.12.7.1.4. 接続を許可するためのハイパーバイザーシステムの設定

開発環境で作業している場合は、環境内の各種仮想ネットワークを介したさまざまなタイプの接続を許可するようにハイパーバイザーシステムを設定します。

注: 実稼働環境で作業している場合は、安全な環境を維持するために、**firewalld** サービスの適切なルールを確立し、SELinux ポリシーを設定する必要があります。

- SELinux の場合は、次のコマンドを入力します。

```
sed -i s/^SELINUX=.*$/SELINUX=permissive/ /etc/selinux/config; setenforce 0
```

- **firewalld** の場合は、次のコマンドを入力します。

```
systemctl disable --now firewalld
```

- **libvirtd** の場合は、以下のコマンドを入力します。

```
systemctl restart libvirtd
```

```
systemctl enable --now libvirtd
```

次に、環境に合わせて DNS を設定します。

1.7.12.7.1.5. 関連情報

- **kcli**, の詳細は、公式の kcli ドキュメント を参照してください。

1.7.12.7.2. デュアルスタックネットワーク用の DNS の設定

DNS の設定は、仮想環境とベアメタル環境の両方で、オフライン環境とオンライン環境の両方で必須です。仮想環境とベアメタル環境の主な違いは、リソースを設定する場所にあります。仮想以外の環境では、**dnsmasq** のような軽量のソリューションではなく、Bind のようなソリューションを使用してください。

- デュアルスタックで IPv6 ネットワークの DNS を設定するには、デフォルトの ingress と DNS の動作 を参照してください。

- ベアメタル上のデュアルスタックネットワーク用に DNS を設定するには、ベアメタル上の DNS の設定 を参照してください。

次にレジストリーをデプロイします。

1.7.12.7.3. デュアルスタックネットワーク用のレジストリーのデプロイ

開発環境の場合は、Podman コンテナーを使用して、小規模な自己ホスト型レジストリーをデプロイします。実稼働環境では、Red Hat Quay、Nexus、Artifactory などのエンタープライズでホストされるレジストリーをデプロイします。

Podman を使用して小規模なレジストリーをデプロイするには、以下の手順を実行します。

1. 特権ユーザーとして **${HOME}** ディレクトリーにアクセスし、次のスクリプトを作成します。

```
#!/usr/bin/env bash

set -euo pipefail

PRIMARY_NIC=$(ls -1 /sys/class/net | grep -v podman | head -1)
export PATH=/root/bin:$PATH
export PULL_SECRET="/root/baremetal/hub/openshift_pull.json" 1

if [[ ! -f $PULL_SECRET ]];then
  echo "Pull Secret not found, exiting..."
  exit 1
fi

dnf -y install podman httpd httpd-tools jq skopeo libseccomp-devel
export IP=$(ip -o addr show $PRIMARY_NIC | head -1 | awk '{print $4}' | cut -d'/' -f1)
REGISTRY_NAME=registry.$(hostname --long)
REGISTRY_USER=dummy
REGISTRY_PASSWORD=dummy
KEY=$(echo -n $REGISTRY_USER:$REGISTRY_PASSWORD | base64)
echo "{\"auths\": {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\", \"email\": \"sample-
email@domain.ltd\"}}}" > /root/disconnected_pull.json
mv ${PULL_SECRET} /root/openshift_pull.json.old
jq ".auths += {\"$REGISTRY_NAME:5000\": {\"auth\": \"$KEY\",\"email\": \"sample-
email@domain.ltd\"}}" < /root/openshift_pull.json.old > $PULL_SECRET
mkdir -p /opt/registry/{auth,certs,data,conf}
cat <<EOF > /opt/registry/conf/config.yml
version: 0.1
log:
  fields:
    service: registry
storage:
  cache:
    blobdescriptor: inmemory
  filesystem:
    rootdirectory: /var/lib/registry
  delete:
    enabled: true
http:
  addr: :5000
  headers:
    X-Content-Type-Options: [nosniff]
```

```
health:
  storagedriver:
    enabled: true
    interval: 10s
    threshold: 3
compatibility:
  schema1:
    enabled: true
EOF
openssl req -newkey rsa:4096 -nodes -sha256 -keyout /opt/registry/certs/domain.key -x509 -
days 3650 -out /opt/registry/certs/domain.crt -subj "/C=US/ST=Madrid/L=San
Bernardo/O=Karmalabs/OU=Guitar/CN=$REGISTRY_NAME" -addext
"subjectAltName=DNS:$REGISTRY_NAME"
cp /opt/registry/certs/domain.crt /etc/pki/ca-trust/source/anchors/
update-ca-trust extract
htpasswd -bBc /opt/registry/auth/htpasswd $REGISTRY_USER $REGISTRY_PASSWORD
podman create --name registry --net host --security-opt label=disable --replace -v
/opt/registry/data:/var/lib/registry:z -v /opt/registry/auth:/auth:z -v
/opt/registry/conf/config.yml:/etc/docker/registry/config.yml -e "REGISTRY_AUTH=htpasswd"
-e "REGISTRY_AUTH_HTPASSWD_REALM=Registry" -e
"REGISTRY_HTTP_SECRET=ALongRandomSecretForRegistry" -e
REGISTRY_AUTH_HTPASSWD_PATH=/auth/htpasswd -v /opt/registry/certs:/certs:z -e
REGISTRY_HTTP_TLS_CERTIFICATE=/certs/domain.crt -e
REGISTRY_HTTP_TLS_KEY=/certs/domain.key docker.io/library/registry:latest
[ "$?" == "0" ] || !!
systemctl enable --now registry
```

1 **PULL_SECRET** の場所は、設定に適した場所に置き換えます。

2. スクリプトファイル **registry.sh** という名前を指定して保存します。スクリプトを実行すると、以下の情報がプルされます。
   - ハイパーバイザーのホスト名に基づくレジストリー名
   - 必要な認証情報およびユーザーアクセスの詳細
3. 次のように実行フラグを追加して、パーミッションを調整します。

```
chmod u+x ${HOME}/registry.sh
```

4. パラメーターを指定せずにスクリプトを実行するには、以下のコマンドを入力します。

```
${HOME}/registry.sh
```

   このスクリプトはサーバーを起動します。

5. このスクリプトは、管理目的で **systemd** サービスを使用します。スクリプトを管理する必要がある場合は、以下のコマンドを使用できます。

```
systemctl status
```

```
systemctl start
```

```
systemctl stop
```

–

レジストリーのルートフォルダーは **/opt/registry** ディレクトリー内にあり、次のサブディレクトリーが含まれています。

- **certs** には TLS 証明書が含まれます。
- **auth** には認証情報が含まれます。
- **data** にはレジストリーイメージが含まれます。
- **conf** にはレジストリー設定が含まれています。

1.7.12.7.4. デュアルスタックネットワークの管理クラスターの設定

OpenShift Container Platform 管理クラスターを設定するには、dev-scripts を使用できます。または、仮想マシンをベースにしている場合は、**kcli** ツールを使用できます。以下は、**kcli** ツールに固有のものです。

1. ハイパーバイザーで使用するために適切なネットワークの準備が完了していることを確認します。ネットワークは、管理クラスターとホステッドクラスターの両方をホストします。以下の **kcli** コマンドを入力します。

   ```
   kcli create network -c 192.168.126.0/24 -P dhcp=false -P dns=false -d 2620:52:0:1306::0/64
   --domain dns.base.domain.name --nodhcp dual
   ```

   ここでは、以下のようになります。

   - **-c** は、ネットワークの CIDR を指定します。
   - **-p dhcp=false** は、設定した **dnsmasq** によって処理される DHCP を無効にするようにネットワークを設定します。
   - **-P dns=false は、** DNS を無効にするようにネットワークを設定します。これも、設定した **dnsmasq** によって処理されます。
   - **--domain** は、検索するドメインを設定します。
   - **dns.base.domain.name** は DNS ベースドメイン名です。
   - **dual** は、作成するネットワークの名前です。

2. ネットワークを作成したら、以下の出力を確認します。

   ```
   [root@hypershiftbm ~]# kcli list network
   Listing Networks...
   +---------+--------+---------------------+-------+------------------+------+
   | Network |  Type  |         Cidr        |  Dhcp |      Domain      | Mode |
   +---------+--------+---------------------+-------+------------------+------+
   | default | routed |   192.168.122.0/24  |  True |     default      | nat  |
   | ipv4    | routed | 2620:52:0:1306::/64 | False | dns.base.domain.name | nat  |
   | ipv4    | routed |   192.168.125.0/24  | False | dns.base.domain.name | nat  |
   | ipv6    | routed | 2620:52:0:1305::/64 | False | dns.base.domain.name | nat  |
   +---------+--------+---------------------+-------+------------------+------+
   ```

   ```
   [root@hypershiftbm ~]# kcli info network ipv6
   Providing information about network ipv6...
   ```

```
cidr: 2620:52:0:1306::/64
dhcp: false
domain: dns.base.domain.name
mode: nat
plan: kvirt
type: routed
```

3. OpenShift Container Platform 管理クラスターをデプロイできるように、プルシークレットと **kcli** プランファイルが配置されていることを確認します。

   a. プルシークレットが **kcli** プランと同じフォルダーにあり、プルシークレットファイルの名前が **openshift_pull.json** であることを確認します。

   b. OpenShift Container Platform 定義を含む **kcli** プランを **mgmt-compact-hub-dual.yaml** ファイルに追加します。ご使用の環境に合わせてファイルの内容を更新してください。

```
plan: hub-dual
force: true
version: stable
tag: "4.x.y-x86_64" (1)
cluster: "hub-dual"
dualstack: true
domain: dns.base.domain.name
api_ip: 192.168.126.10
ingress_ip: 192.168.126.11
service_networks:
- 172.30.0.0/16
- fd02::/112
cluster_networks:
- 10.132.0.0/14
- fd01::/48
disconnected_url: registry.dns.base.domain.name:5000
disconnected_update: true
disconnected_user: dummy
disconnected_password: dummy
disconnected_operators_version: v4.14
disconnected_operators:
- name: metallb-operator
- name: lvms-operator
  channels:
  - name: stable-4.13
disconnected_extra_images:
- quay.io/user-name/trbsht:latest
- quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
- registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
dualstack: true
disk_size: 200
extra_disks: [200]
memory: 48000
numcpus: 16
ctlplanes: 3
workers: 0
manifests: extra-manifests
metal3: true
network: dual
users_dev: developer
```

```
users_devpassword: developer
users_admin: admin
users_adminpassword: admin
metallb_pool: dual-virtual-network
metallb_ranges:
- 192.168.126.150-192.168.126.190
metallb_autoassign: true
apps:
- users
- lvms-operator
- metallb-operator
vmrules:
- hub-bootstrap:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:07
- hub-ctlplane-0:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:01
- hub-ctlplane-1:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:02
- hub-ctlplane-2:
    nets:
    - name: ipv6
      mac: aa:aa:aa:aa:10:03
```

1 **4.x.y** を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

4. 管理クラスターをプロビジョニングするには、以下のコマンドを入力します。

```
kcli create cluster openshift --pf mgmt-compact-hub-dual.yaml
```

次に、Web サーバーを設定します。

1.7.12.7.4.1. 関連情報

- **kcli** プランファイルのパラメーターの詳細は、**kcli** 公式ドキュメントの parameters.yml の作成 を参照してください。

1.7.12.7.5. デュアルスタックネットワーク用の Web サーバーの設定

ホステッドクラスターとしてデプロイする OpenShift Container Platform リリースに関連付けられた Red Hat Enterprise Linux CoreOS (RHCOS) イメージをホストするには、追加の Web サーバーを設定する必要があります。

Web サーバーを設定するには、以下の手順を実行します。

1. 以下のコマンドを入力して、使用する OpenShift Container Platform リリースから **openshift-install** バイナリーを展開します。

```
oc adm -a ${LOCAL_SECRET_JSON} release extract --command=openshift-install
"${LOCAL_REGISTRY}/${LOCAL_REPOSITORY}:${OCP_RELEASE}-${ARCHITECTURE}"
```

2. 次のスクリプトを実行します。このスクリプトは、**/opt/srv** ディレクトリーにフォルダーを作成します。このフォルダーには、ワーカーノードをプロビジョニングするための RHCOS イメージが含まれています。

```
#!/bin/bash

WEBSRV_FOLDER=/opt/srv
ROOTFS_IMG_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.pxe.rootfs.location')" 1
LIVE_ISO_URL="$(./openshift-install coreos print-stream-json | jq -r
'.architectures.x86_64.artifacts.metal.formats.iso.disk.location')" 2

mkdir -p ${WEBSRV_FOLDER}/images
curl -Lk ${ROOTFS_IMG_URL} -o
${WEBSRV_FOLDER}/images/${ROOTFS_IMG_URL##*/}
curl -Lk ${LIVE_ISO_URL} -o ${WEBSRV_FOLDER}/images/${LIVE_ISO_URL##*/}
chmod -R 755 ${WEBSRV_FOLDER}/*

## Run Webserver
podman ps --noheading | grep -q websrv-ai
if [[ $? == 0 ]];then
    echo "Launching Registry pod..."
    /usr/bin/podman run --name websrv-ai --net host -v /opt/srv:/usr/local/apache2/htdocs:z
quay.io/alosadag/httpd:p8080
fi
```

1 **ROOTFS_IMG_URL** 値は OpenShift CI Release ページにあります。

2 **LIVE_ISO_URL** 値は、OpenShift CI リリースページで確認できます。

ダウンロードが完了すると、コンテナーが実行され、Web サーバー上でイメージをホストします。このコンテナーは公式 HTTPd イメージのバリエーションを使用しているので、IPv6 ネットワークでの動作も可能になります。

1.7.12.7.6. デュアルスタックネットワークのイメージミラーリングの設定

イメージミラーリングは、**registry.redhat.com** や **quay.io** などの外部レジストリーからイメージを取得し、プライベートレジストリーに保存するプロセスです。

1.7.12.7.6.1. ミラーリングプロセスの完了

注: ミラーリングプロセスは、レジストリーサーバーの実行後に開始してください。

次の手順では、**ImageSetConfiguration** オブジェクトを使用するバイナリーである、**oc-mirror** ツールが使用されます。このファイルで、以下の情報を指定できます。

- ミラーリングする OpenShift Container Platform バージョン。バージョンは **quay.io** にあります。
- ミラーリングする追加の Operator。パッケージは個別に選択します。

- リポジトリーに追加する追加のイメージ。

イメージのミラーリングを設定するには、以下の手順を実行します。

1. **${HOME}/.docker/config.json** ファイルが、ミラーリング元のレジストリーとイメージのプッシュ先のプライベートレジストリーで更新されていることを確認します。

2. 次の例を使用して、ミラーリングに使用する **ImageSetConfiguration** オブジェクトを作成します。環境に合わせて必要に応じて値を置き換えます。

```
apiVersion: mirror.openshift.io/v1alpha2
kind: ImageSetConfiguration
storageConfig:
  registry:
    imageURL: registry.dns.base.domain.name:5000/openshift/release/metadata:latest
mirror:
  platform:
    channels:
    - name: candidate-4.14
      minVersion: 4.x.y-x86_64 (1)
      maxVersion: 4.x.y-x86_64
      type: ocp
    graph: true
  additionalImages:
  - name: quay.io/karmab/origin-keepalived-ipfailover:latest
  - name: quay.io/karmab/kubectl:latest
  - name: quay.io/karmab/haproxy:latest
  - name: quay.io/karmab/mdns-publisher:latest
  - name: quay.io/karmab/origin-coredns:latest
  - name: quay.io/karmab/curl:latest
  - name: quay.io/karmab/kcli:latest
  - name: quay.io/user-name/trbsht:latest
  - name: quay.io/user-name/hypershift:BMSelfManage-v4.14-rc-v3
  - name: registry.redhat.io/openshift4/ose-kube-rbac-proxy:v4.10
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.14
    packages:
    - name: lvms-operator
    - name: local-storage-operator
    - name: odf-csi-addons-operator
    - name: odf-operator
    - name: mcg-operator
    - name: ocs-operator
    - name: metallb-operator
```

(1) **4.x.y** を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

3. 次のコマンドを入力して、ミラーリングプロセスを開始します。

```
oc-mirror --source-skip-tls --config imagesetconfig.yaml docker://${REGISTRY}
```

ミラーリングプロセスが完了すると、**oc-mirror-workspace/results-XXXXXX/** という名前の新しいフォルダーが作成されます。このフォルダーには、ICSP と、ホステッドクラスターに適用するカタログソースが含まれます。

4. **oc adm release Mirror** コマンドを使用して、OpenShift Container Platform の夜間バージョンまたは CI バージョンをミラーリングします。以下のコマンドを入力します。

```
REGISTRY=registry.$(hostname --long):5000

oc adm release mirror \
  --from=registry.ci.openshift.org/ocp/release:4.x.y-x86_64 \
  --to=${REGISTRY}/openshift/release \
  --to-release-image=${REGISTRY}/openshift/release-
images:registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64
```

   4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

5. 非接続ネットワークへのインストール の手順に従って、最新の multicluster engine Operator イメージをミラーリングします。

1.7.12.7.6.2. 管理クラスターでのオブジェクトの適用

ミラーリングプロセスが完了したら、管理クラスターに 2 つのオブジェクトを適用する必要があります。

- イメージコンテンツソースポリシー (ICSP) またはイメージダイジェストミラーセット (IDMS)
- カタログソース

oc-mirror ツールを使用すると、出力アーティファクトは **oc-mirror-workspace/results-XXXXXX/** という名前のフォルダーに保存されます。

ICSP または IDMS は、ノードを再起動せずに、各ノードで kubelet を再起動する **MachineConfig** 変更を開始します。ノードが **READY** としてマークされたら、新しく生成されたカタログソースを適用する必要があります。

カタログソースは、カタログイメージのダウンロードや処理を行い、そのイメージに含まれるすべての **packagemanifests** を取得するなど、**openshift-marketplace** Operator でアクションを開始します。

1. 新しいソースを確認するには、新しい **CatalogSource** をソースとして使用して次のコマンドを実行します。

```
oc get packagemanifest
```

2. アーティファクトを適用するには、次の手順を実行します。

   a. 次のコマンドを入力して、ICSP または IDMS アーティファクトを作成します。

```
oc apply -f oc-mirror-workspace/results-XXXXXX/imageContentSourcePolicy.yaml
```

   b. ノードの準備が完了するまで待ってから、次のコマンドを入力します。

```
oc apply -f catalogSource-XXXXXXXX-index.yaml
```

次に、multicluster engine operator をデプロイします。

1.7.12.7.6.3. 関連情報

- 仮想環境で作業している場合は、ミラーリングを設定した後、OpenShift Virtualization 上のホストされたコントロールプレーンの前提条件 を満たしていることを確認してください。
- OpenShift Container Platform の夜間ミラーリングまたは CI バージョンのミラーリングの詳細は、oc-mirror プラグインを使用した非接続インストールのイメージのミラーリング を参照してください。

1.7.12.7.7. デュアルスタックネットワーク用の multicluster engine Operator のデプロイ

multicluster engine Operator は、プロバイダー間でクラスターをデプロイメントする場合に重要な役割を果たします。Red Hat Advanced Cluster Management をすでにインストールしている場合は、multicluster engine Operator は自動的にインストールされるため、インストールする必要はありません。

multicluster engine Operator がインストールされていない場合は、次のドキュメントを参照して、前提条件とインストール手順を確認してください。

- multicluster engine operator を使用したクラスターライフサイクルについて
- multicluster engine operator のインストールとアップグレード

1.7.12.7.7.1. AgentServiceConfig リソースのデプロイ

AgentServiceConfig カスタムリソースは、multicluster engine operator の一部である Assisted Service アドオンの重要なコンポーネントです。このコンポーネントは、ベアメタルクラスターをデプロイメントします。アドオンが有効な場合に、**AgentServiceConfig** リソースをデプロイしてアドオンを設定します。

AgentServiceConfig リソースの設定に加えて、multicluster engine Operator が非接続環境で適切に機能するように、追加の config map を含める必要があります。

1. 次の config map を追加してカスタムレジストリーを設定します。これには、デプロイメントをカスタマイズするための非接続環境の情報が含まれています。

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: custom-registries
  namespace: multicluster-engine
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
  registries.conf: |
    unqualified-search-registries = ["registry.access.redhat.com", "docker.io"]

    [[registry]]
    prefix = ""
    location = "registry.redhat.io/openshift4"
    mirror-by-digest-only = true

    [[registry.mirror]]
      location = "registry.dns.base.domain.name:5000/openshift4" 1
```

```
    [[registry]]
    prefix = ""
    location = "registry.redhat.io/rhacm2"
    mirror-by-digest-only = true
    ...
    ...
```

1 **dns.base.domain.name** は DNS ベースドメイン名に置き換えます。

オブジェクトには、以下の 2 つのフィールドが含まれます。

- カスタム CA: このフィールドには、デプロイメントのさまざまなプロセスに読み込まれる認証局 (CA) が含まれます。
- レジストリー: **Registries.conf** フィールドには、元のソースレジストリーではなくミラーレジストリーから使用する必要があるイメージと namespace に関する情報が含まれています。

2. 次の例に示すように、**AssistedServiceConfig** オブジェクトを追加して、Assisted Service を設定します。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: AgentServiceConfig
metadata:
  annotations:
    unsupported.agent-install.openshift.io/assisted-service-configmap: assisted-service-config
1
  name: agent
  namespace: multicluster-engine
spec:
  mirrorRegistryRef:
    name: custom-registries 2
  databaseStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 10Gi
  filesystemStorage:
    storageClassName: lvms-vg1
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: 20Gi
  osImages: 3
  - cpuArchitecture: x86_64
    openshiftVersion: "4.14"
    rootFSUrl: http://registry.dns.base.domain.name:8080/images/rhcos-
414.92.202308281054-0-live-rootfs.x86_64.img 4
```

```
    url: http://registry.dns.base.domain.name:8080/images/rhcos-414.92.202308281054-0-
live.x86_64.iso
    version: 414.92.202308281054-0
```

1. **metadata.annotations"unsupported.agent-install.openshift.io/assisted-service-configmap"** アノテーションは、Operator が動作をカスタマイズするために使用する config map 名を参照します。
2. **spec.mirrorRegistryRef.name** アノテーションは、Assisted Service Operator が使用する非接続のレジストリー情報を含む config map を指します。この config map は、デプロイメントプロセス中にこれらのリソースを追加します。
3. **spec.osImages** フィールドには、この Operator がデプロイできるさまざまなバージョンが含まれています。このフィールドは必須です。この例では、**RootFS** ファイルと **LiveISO** ファイルがすでにダウンロードされていることを前提としています。
4. **rootFSUrl フィールド** と **url** フィールドで、**dns.base.domain.name** を DNS ベースドメイン名に置き換えます。

3. すべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに適用し、これらのオブジェクトをデプロイします。起動するには、以下のコマンドを実行します。

```
oc apply -f agentServiceConfig.yaml
```

このコマンドは、次の出力例に示すように、2 つの Pod をトリガーします。

```
assisted-image-service-0                  1/1     Running   2          11d (1)
assisted-service-668b49548-9m7xw          2/2     Running   5          11d (2)
```

1. **Assisted-image-service** Pod は、デプロイするクラスターごとにカスタマイズされた、Red Hat Enterprise Linux CoreOS (RHCOS) 起動イメージテンプレートを作成します。
2. **assisted-service** は Operator を参照します。

1.7.12.7.8. デュアルスタックネットワークの TLS 証明書の設定

オフライン環境で Hosted control plane を設定するプロセスに、いくつかの TLS 証明書が関与します。認証局 (CA) を管理クラスターに追加するには、OpenShift Container Platform コントロールプレーンおよびワーカーノード内の以下のファイルの内容を変更する必要があります。

- **/etc/pki/ca-trust/extracted/pem/**
- **/etc/pki/ca-trust/source/anchors**
- **/etc/pki/tls/certs/**

CA を管理クラスターに追加するには、次の手順を実行します。

1. OpenShift Container Platform の公式ドキュメントの CA バンドルの更新 の手順を完了します。この方法には、CA を OpenShift Container Platform ノードにデプロイする **image-registry-operator** の使用が含まれます。
2. この方法が実際の状況に該当しない場合は、管理クラスター内の **openshift-config** namespace に **user-ca-bundle** という名前の config map が含まれているかどうかを確認してください。

- namespace にその config map が含まれている場合は、次のコマンドを入力します。

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt

oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${REGISTRY_CERT_PATH}
```

- namespace にその config map が含まれていない場合は、以下のコマンドを入力します。

```
## REGISTRY_CERT_PATH=<PATH/TO/YOUR/CERTIFICATE/FILE>
export REGISTRY_CERT_PATH=/opt/registry/certs/domain.crt
export TMP_FILE=$(mktemp)

oc get cm -n openshift-config user-ca-bundle -ojsonpath='{.data.ca-bundle\.crt}' >
${TMP_FILE}
echo >> ${TMP_FILE}
echo \#registry.$(hostname --long) >> ${TMP_FILE}
cat ${REGISTRY_CERT_PATH} >> ${TMP_FILE}
oc create configmap user-ca-bundle -n openshift-config --from-file=ca-
bundle.crt=${TMP_FILE} --dry-run=client -o yaml | kubectl apply -f -
```

1.7.12.7.9. デュアルスタックネットワーク用のホステッドクラスターのデプロイ

ホステッドクラスターは、管理クラスターにホストされたコントロールプレーンと API エンドポイントを備えた OpenShift Container Platform クラスターです。ホストされたクラスターには、コントロールプレーンとそれに対応するデータプレーンが含まれます。

Red Hat Advanced Cluster Management のコンソールを使用してホステッドクラスターを作成できますが、次の手順ではマニフェストを使用するため、関連するアーティファクトをより柔軟に変更できます。

1.7.12.7.9.1. ホステッドクラスターオブジェクトのデプロイ

この手順では、次の値が使用されます。

- HostedCluster name: **hosted-dual**
- HostedCluster namespace: **clusters**
- Disconnected: **true**
- Network stack: **Dual**

通常、HyperShift Operator は **HostedControlPlane** namespace を作成します。ただし、この場合は、HyperShift Operator が **HostedCluster** オブジェクトの調整を開始する前に、すべてのオブジェクトを含める必要があります。その後、Operator が調整プロセスを開始すると、所定の場所にあるすべてのオブジェクトを見つけることができます。

1. namespace に関する次の情報を含めて、YAML ファイルを作成します。

```
---
apiVersion: v1
kind: Namespace
metadata:
```

```
  creationTimestamp: null
  name: clusters-hosted-dual
spec: {}
status: {}
---
apiVersion: v1
kind: Namespace
metadata:
  creationTimestamp: null
  name: clusters
spec: {}
status: {}
```

2. config map とシークレットに関する次の情報を含む YAML ファイルを作成し、**HostedCluster** デプロイメントに追加します。

```
---
apiVersion: v1
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    -----END CERTIFICATE-----
kind: ConfigMap
metadata:
  name: user-ca-bundle
  namespace: clusters
---
apiVersion: v1
data:
  .dockerconfigjson: xxxxxxxxx
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-dual-pull-secret
  namespace: clusters
---
apiVersion: v1
kind: Secret
metadata:
  name: sshkey-cluster-hosted-dual
  namespace: clusters
stringData:
  id_rsa.pub: ssh-rsa xxxxxxxxx
---
apiVersion: v1
data:
  key: nTPtVBEt03owkrKhIdmSW8jrWRxU57KO/fnZa8oaG0Y=
kind: Secret
metadata:
  creationTimestamp: null
  name: hosted-dual-etcd-encryption-key
  namespace: clusters
type: Opaque
```

3. RBAC ロールを含む YAML ファイルを作成し、Assisted Service エージェントが Hosted

3. RBAC ロールを含む YAML ファイルを作成し、Assisted Service エージェントが Hosted control plane と同じ **HostedControlPlane** namespace に配置し、引き続きクラスター API で管理されるようにします。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  creationTimestamp: null
  name: capi-provider-role
  namespace: clusters-hosted-dual
rules:
- apiGroups:
  - agent-install.openshift.io
  resources:
  - agents
  verbs:
  - '*'
```

4. **HostedCluster** オブジェクトに関する情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: HostedCluster
metadata:
  name: hosted-dual
  namespace: clusters
spec:
  additionalTrustBundle:
    name: "user-ca-bundle"
  olmCatalogPlacement: guest
  imageContentSources: (1)
  - source: quay.io/openshift-release-dev/ocp-v4.0-art-dev
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release (2)
  - source: quay.io/openshift-release-dev/ocp-release
    mirrors:
    - registry.dns.base.domain.name:5000/openshift/release-images
  - mirrors:
  ...
  ...
  autoscaling: {}
  controllerAvailabilityPolicy: SingleReplica
  dns:
    baseDomain: dns.base.domain.name
  etcd:
    managed:
      storage:
        persistentVolume:
          size: 8Gi
        restoreSnapshotURL: null
        type: PersistentVolume
    managementType: Managed
  fips: false
  networking:
    clusterNetwork:
    - cidr: 10.132.0.0/14
```

```
    - cidr: fd01::/48
    networkType: OVNKubernetes
    serviceNetwork:
    - cidr: 172.31.0.0/16
    - cidr: fd02::/112
  platform:
    agent:
      agentNamespace: clusters-hosted-dual
    type: Agent
  pullSecret:
    name: hosted-dual-pull-secret
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64 (3)
  secretEncryption:
    aescbc:
      activeKey:
        name: hosted-dual-etcd-encryption-key
    type: aescbc
  services:
  - service: APIServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: OAuthServer
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: OIDC
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: Konnectivity
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  - service: Ignition
    servicePublishingStrategy:
      nodePort:
        address: api.hosted-dual.dns.base.domain.name
      type: NodePort
  sshKey:
    name: sshkey-cluster-hosted-dual
status:
  controlPlaneEndpoint:
    host: ""
    port: 0
```

(1) **imageContentSources** セクションには、ホステッドクラスター内のユーザーワークロードのミラー参照が含まれます。

(2) YAML ファイル全体で、**dns.base.domain.name** は、DNS ベースドメイン名に置き換えます。

3 **4.x.y** を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

5. OpenShift Container Platform リリースの HyperShift Operator リリースを指すアノテーションを **HostedCluster** オブジェクトに追加します。

   a. 次のコマンドを入力して、イメージペイロードを取得します。

   ```
   oc adm release info registry.dns.base.domain.name:5000/openshift-release-dev/ocp-release:4.x.y-x86_64 | grep hypershift
   ```

   ここで、**dns.base.domain.name** は DNS ベースドメイン名であり、**4.x.y** は使用するサポートされている OpenShift Container Platform のバージョンです。

   b. 以下の出力を参照してください。

   ```
   hypershift
   sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   c. OpenShift Container Platform Images namespace を使用して、次のコマンドを入力してダイジェストを確認します。

   ```
   podman pull registry.dns.base.domain.name:5000/openshift-release-dev/ocp-v4.0-art-dev@sha256:31149e3e5f8c5e5b5b100ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   ```

   dns.base.domain.name は DNS ベースドメイン名です。

   d. 以下の出力を参照してください。

   ```
   podman pull
   registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
   ff2d89975cf5f7a73801b2c06c639bf6648766117f8
   Trying to pull
   registry.dns.base.domain.name:5000/openshift/release@sha256:31149e3e5f8c5e5b5b100
   ff2d89975cf5f7a73801b2c06c639bf6648766117f8...
   Getting image source signatures
   Copying blob d8190195889e skipped: already exists
   Copying blob c71d2589fba7 skipped: already exists
   Copying blob d4dc6e74b6ce skipped: already exists
   Copying blob 97da74cc6d8f skipped: already exists
   Copying blob b70007a560c9 done
   Copying config 3a62961e6e done
   Writing manifest to image destination
   Storing signatures
   3a62961e6ed6edab46d5ec8429ff1f41d6bb68de51271f037c6cb8941a007fde
   ```

   注: HostedCluster オブジェクトに設定されるリリースイメージでは、タグではなくダイジェストを使用する必要があります (例: **quay.io/openshift-release-dev/ocp-release@sha256:e3ba11bd1e5e8ea5a0b36a75791c90f29afb0fdbe4125be4e48f69c76a5c47a0**)。

6. YAML ファイルで定義したすべてのオブジェクトを 1 つのファイルに連結し、管理クラスターに対して適用して作成します。起動するには、以下のコマンドを実行します。

```
oc apply -f 01-4.14-hosted_cluster-nodeport.yaml
```

7. Hosted control plane の出力を参照してください。

```
NAME                                          READY   STATUS    RESTARTS   AGE
capi-provider-5b57dbd6d5-pxlqc                1/1     Running   0          3m57s
catalog-operator-9694884dd-m7zzv              2/2     Running   0          93s
cluster-api-f98b9467c-9hfrq                   1/1     Running   0          3m57s
cluster-autoscaler-d7f95dd5-d8m5d             1/1     Running   0          93s
cluster-image-registry-operator-5ff5944b4b-648ht   1/2     Running   0          93s
cluster-network-operator-77b896ddc-wpkq8      1/1     Running   0          94s
cluster-node-tuning-operator-84956cd484-4hfgf 1/1     Running   0          94s
cluster-policy-controller-5fd8595d97-rhbwf    1/1     Running   0          95s
cluster-storage-operator-54dcf584b5-xrnts     1/1     Running   0          93s
cluster-version-operator-9c554b999-l22s7      1/1     Running   0          95s
control-plane-operator-6fdc9c569-t7hr4        1/1     Running   0          3m57s
csi-snapshot-controller-785c6dc77c-8ljmr      1/1     Running   0          77s
csi-snapshot-controller-operator-7c6674bc5b-d9dtp   1/1     Running   0          93s
csi-snapshot-webhook-5b8584875f-2492j         1/1     Running   0          77s
dns-operator-6874b577f-9tc6b                  1/1     Running   0          94s
etcd-0                                        3/3     Running   0          3m39s
hosted-cluster-config-operator-f5cf5c464-4nmbh   1/1     Running   0          93s
ignition-server-6b689748fc-zdqzk              1/1     Running   0          95s
ignition-server-proxy-54d4bb9b9b-6zkg7        1/1     Running   0          95s
ingress-operator-6548dc758b-f9gtg             1/2     Running   0          94s
konnectivity-agent-7767cdc6f5-tw782           1/1     Running   0          95s
kube-apiserver-7b5799b6c8-9f5bp               4/4     Running   0          3m7s
kube-controller-manager-5465bc4dd6-zpdlk      1/1     Running   0          44s
kube-scheduler-5dd5f78b94-bbbck               1/1     Running   0          2m36s
machine-approver-846c69f56-jxvfr              1/1     Running   0          92s
oauth-openshift-79c7bf44bf-j975g              2/2     Running   0          62s
olm-operator-767f9584c-4lcl2                  2/2     Running   0          93s
openshift-apiserver-5d469778c6-pl8tj          3/3     Running   0          2m36s
openshift-controller-manager-6475fdff58-hl4f7   1/1     Running   0          95s
openshift-oauth-apiserver-dbbc5cc5f-98574     2/2     Running   0          95s
openshift-route-controller-manager-5f6997b48f-s9vdc   1/1     Running   0          95s
packageserver-67c87d4d4f-kl7qh                2/2     Running   0          93s
```

8. ホステッドクラスターの出力を確認します。

```
NAMESPACE   NAME          VERSION   KUBECONFIG                PROGRESS   AVAILABLE   PROGRESSING   MESSAGE
clusters    hosted-dual             hosted-admin-kubeconfig   Partial    True        False         The hosted control plane is available
```

次に、**NodePool** オブジェクトを作成します。

1.7.12.7.9.2. ホステッドクラスターの NodePool オブジェクトの作成

NodePool は、ホステッドクラスターに関連付けられたスケーラブルなワーカーノードのセットです。**NodePool** マシンアーキテクチャーは特定のプール内で一貫性を保ち、コントロールプレーンのマシンアーキテクチャーから独立しています。

1. **NodePool** オブジェクトに関する次の情報を含む YAML ファイルを作成し、必要に応じて値を置き換えます。

```
apiVersion: hypershift.openshift.io/v1beta1
kind: NodePool
metadata:
  creationTimestamp: null
  name: hosted-dual
  namespace: clusters
spec:
  arch: amd64
  clusterName: hosted-dual
  management:
    autoRepair: false (1)
    upgradeType: InPlace (2)
  nodeDrainTimeout: 0s
  platform:
    type: Agent
  release:
    image: registry.dns.base.domain.name:5000/openshift/release-images:4.x.y-x86_64 (3)
  replicas: 0
status:
  replicas: 0 (4)
```

(1) ノードが削除された場合、ノードは再作成されないため、**autoRepair** フィールドは **false** に設定されます。

(2) **upgradeType** は **InPlace** に設定されます。これは、アップグレード中に同じベアメタルノードが再利用されることを示します。

(3) この **NodePool** に含まれるすべてのノードは、OpenShift Container Platform バージョン **4.x.y-x86_64** に基づいています。**dns.base.domain.name** の値を DNS ベースドメイン名に置き換え、**4.x.y** の値を、使用するサポートされている OpenShift Container Platform のバージョンに置き換えます。

(4) **replicas** の値は **0** に設定されているため、必要に応じてスケールを変更できます。すべての手順が完了するまで、**NodePool** レプリカを 0 に保つことが重要です。

2. 次のコマンドを入力して、**NodePool** オブジェクトを作成します。

```
oc apply -f 02-nodepool.yaml
```

3. 出力を参照してください。

```
NAMESPACE   NAME         CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted-dual   hosted    0                          False        False        4.x.y-x86_64
```

次に、**InfraEnv** リソースを作成します。

1.7.12.7.9.3. ホステッドクラスターの InfraEnv リソースの作成

InfraEnv リソースは、**pullSecretRef** や **sshAuthorizedKey** などの重要な詳細を含む Assisted Service オブジェクトです。これらの詳細は、ホステッドクラスター用にカスタマイズされた Red Hat Enterprise Linux CoreOS (RHCOS) ブートイメージを作成するために使用されます。

1. **InfraEnv** リソースに関する次の情報を含めて YAML ファイルを作成し、必要に応じて値を置き換えます。

```
---
apiVersion: agent-install.openshift.io/v1beta1
kind: InfraEnv
metadata:
  name: hosted-dual
  namespace: clusters-hosted-dual
spec:
  pullSecretRef: (1)
    name: pull-secret
  sshAuthorizedKey: ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQDk7ICaUE+/k4zTpxLk4+xFdHi4ZuDi5qjeF52afsNk
w0w/glILHhwpL5gnp5WkRuL8GwJuZ1VqLC9EKrdmegn4MrmUlq7WTsP0VFOZFBfq2XRUxo
1wrRdor2z0Bbh93ytR+ZsDbbLlGngXaMa0Vbt+z74FqlcajbHTZ6zBmTpBVq5RHtDPgKITdpE1f
ongp7+ZXQNBlkaavaqv8bnyrP4BWahLP4iO9/xJF9lQYboYwEEDzmnKLMW1VtCE6nJzEgWC
ufACTbxpNS7GvKtoHT/OVzw8ArEXhZXQUS1UY8zKsX2iXwmyhw5Sj6YboA8WICs4z+TrFP8
9LmxXY0j6536TQFyRz1iB4WWvCbH5n6W+ABV2e8ssJB1AmEy8QYNwpJQJNpSxzoKBjI73X
vPYYC/IjPFMySwZqrSZCkJYqQ023ySkaQxWZT7in4KeMu7eS2tC+Kn4deJ7KwwUycx8n6RH
MeD8Qg9flTHCv3gmab8JKZJqN3hW1D378JuvmIX4V0= (2)
```

(1) **pullSecretRef** は、プルシークレットが使用される **InfraEnv** と同じ namespace 内の config map を参照します。

(2) **sshAuthorizedKey** は、ブートイメージに配置される SSH 公開鍵を表します。SSH 鍵を使用すると、**core** ユーザーとしてワーカーノードにアクセスできます。

2. 次のコマンドを入力して、**InfraEnv** リソースを作成します。

```
oc apply -f 03-infraenv.yaml
```

3. 以下の出力を参照してください。

```
NAMESPACE             NAME     ISO CREATED AT
clusters-hosted-dual  hosted   2023-09-11T15:14:10Z
```

次に、ワーカーノードを作成します。

1.7.12.7.9.4. ホステッドクラスター用のワーカーノードの作成

ベアメタルプラットフォームで作業している場合、**BareMetalHost** の詳細が正しく設定されていることを確認するためにワーカーノードを作成することが重要です。

仮想マシンを使用している場合は、次の手順を実行して、Metal3 Operator が使用する空のワーカーノードを作成できます。これには、**kcli** ツールを使用します。

1. ワーカーノードを作成するのが初めてではない場合は、まず以前の設定を削除する必要があります。これには、次のコマンドを入力してプランを削除します。

```
kcli delete plan hosted-dual
```

   a. プランを削除するかどうかを確認するプロンプトが表示されたら、**y** と入力します。

b. プランが削除されたことを示すメッセージが表示されることを確認します。

2. 次のコマンドを入力して仮想マシンを作成します。

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\":
\"aa:aa:aa:aa:11:01\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1101 -P
name=hosted-dual-worker0
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\":
\"aa:aa:aa:aa:11:02\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1102 -P
name=hosted-dual-worker1
```

```
kcli create vm -P start=False -P uefi_legacy=true -P plan=hosted-dual -P memory=8192 -P
numcpus=16 -P disks=[200,200] -P nets=["{\"name\": \"dual\", \"mac\":
\"aa:aa:aa:aa:11:03\"}"] -P uuid=aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa1103 -P
name=hosted-dual-worker2
```

```
systemctl restart ksushy
```

ここでは、以下のようになります。

- **start=False** は、仮想マシン (VM) が作成時に自動的に起動しないことを意味します。
- **uefi_legacy=true** は、以前の UEFI 実装との互換性を確保するために UEFI レガシーブートを使用することを意味します。
- **plan=hosted-dual** は、マシンのグループをクラスターとして識別するプラン名を示します。
- **memory=8192** および **numcpus=16** は、RAM や CPU などの仮想マシンのリソースを指定するパラメーターです。
- **discs=200,200** は、VM 内に 2 つのシンプロビジョニングディスクを作成していることを示します。
- **nets=[{"name": "dual", "mac": "aa:aa:aa:aa:02:13"}]** は、接続するネットワーク名やプライマリーインターフェイスの MAC アドレスなど、ネットワークの詳細です。
- **restart ksushy** は、**ksushy** ツールを再起動して、追加した VM をツールが確実に検出できるようにします。

3. 結果の出力を確認します。

```
+--------------------+--------+------------------+-------------------------------------------------+-----
--------+---------+
|        Name        | Status |        Ip        |                  Source                  |    Plan    | Profile
|
+--------------------+--------+------------------+-------------------------------------------------+-----
--------+---------+
|    hosted-worker0  |  down  |                  |                                                  | hosted-dual |
kvirt  |
|    hosted-worker1  |  down  |                  |                                                  | hosted-dual |
kvirt  |
```

```
|   hosted-worker2   | down |                |                                                  | hosted-dual |
kvirt |
+--------------------+------+----------------+--------------------------------------------------+-----
--------+---------+
```

次に、ホステッドクラスターのベアメタルホストを作成します。

1.7.12.7.9.5. ホステッドクラスターのベアメタルホスト作成

ベアメタルホスト は、物理的な詳細と論理詳細を含む **openshift-machine-api** オブジェクトで、Metal3 Operator によって識別できるようになっています。これらの詳細は、**agents** と呼ばれる他の Assisted Service オブジェクトに関連付けられています。

重要: ベアメタルホストと移行先ノードを作成する前に、仮想マシンを作成する必要があります。

ベアメタルホストを作成するには、以下の手順を実行します。

1. 次の情報を使用して YAML ファイルを作成します。
   注記: ベアメタルホストの認証情報を保持するシークレットが 1 つ以上あるため、ワーカーノードごとに少なくとも 2 つのオブジェクトを作成する必要があります。

```
---
apiVersion: v1
kind: Secret
metadata:
  name: hosted-dual-worker0-bmc-secret
  namespace: clusters-hosted-dual
data:
  password: YWRtaW4=
  username: YWRtaW4=
type: Opaque
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: hosted-dual-worker0
  namespace: clusters-hosted-dual
  labels:
    infraenvs.agent-install.openshift.io: hosted-dual (1)
  annotations:
    inspect.metal3.io: disabled
    bmac.agent-install.openshift.io/hostname: hosted-dual-worker0 (2)
spec:
  automatedCleaningMode: disabled (3)
  bmc:
    disableCertificateVerification: true (4)
    address: redfish-virtualmedia://[192.168.126.1]:9000/redfish/v1/Systems/local/hosted-dual-
worker0 (5)
    credentialsName: hosted-dual-worker0-bmc-secret (6)
  bootMACAddress: aa:aa:aa:aa:02:11 (7)
  online: true (8)
```

(1) **infraenvs.agent-install.openshift.io** は、Assisted Installer オブジェクトと **BareMetalHost** オブジェクト間のリンクとして機能します。

2 **bmac.agent-install.openshift.io/hostname** は、デプロイメント中に採用されるノード名を表します。

3 **automatedCleaningMode** は、ノードが Metal3 Operator によって消去されるのを防ぎます。

4 **disableCertificateVerification** は **true** に設定され、クライアントから証明書の検証がバイパスされます。

5 **address** は、ワーカーノードのベースボード管理コントローラー (BMC) アドレスを示します。

6 **credentialsName** は、ユーザーとパスワードの認証情報が保存されるシークレットを指します。

7 **bootMACAddress** は、ノードの起動元のインターフェイス MAC アドレスを示します。

8 **online** は、**BareMetalHost** オブジェクトが作成された後のノードの状態を定義します。

2. 次のコマンドを入力して、**BareMetalHost** オブジェクトをデプロイします。

```
oc apply -f 04-bmh.yaml
```

プロセス中に、次の出力が確認できます。

- この出力は、プロセスがノードに到達しようとしていることを示しています。

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted   hosted-worker0   registering             true         2s
clusters-hosted   hosted-worker1   registering             true         2s
clusters-hosted   hosted-worker2   registering             true         2s
```

- この出力は、ノードが起動していることを示しています。

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted   hosted-worker0   provisioning            true         16s
clusters-hosted   hosted-worker1   provisioning            true         16s
clusters-hosted   hosted-worker2   provisioning            true         16s
```

- この出力は、ノードが正常に起動したことを示しています。

```
NAMESPACE        NAME           STATE        CONSUMER   ONLINE   ERROR   AGE
clusters-hosted   hosted-worker0   provisioned             true         67s
clusters-hosted   hosted-worker1   provisioned             true         67s
clusters-hosted   hosted-worker2   provisioned             true         67s
```

3. ノードが起動したら、次の例に示すように、namespace のエージェントに注目してください。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411             true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412             true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413             true       auto-assign
```

エージェントは、インストールに使用できるノードを表します。ホステッドクラスターにノードを割り当てるには、ノードプールをスケールアップします。

1.7.12.7.9.6. ノードプールのスケールアップ

ベアメタルホストを作成すると、そのステータスが **Registering Provisioning**、**Provisioned** に変わります。ノードは、エージェントの **LiveISO** と、**agent** という名前のデフォルトの Pod で始まります。このエージェントは、Assisted Service Operator から OpenShift Container Platform ペイロードをインストールする指示を受け取ります。

1. ノードプールをスケールアップするには、次のコマンドを入力します。

```
oc -n clusters scale nodepool hosted-dual --replicas 3
```

2. スケーリングプロセスが完了すると、エージェントがホステッドクラスターに割り当てられていることがわかります。

```
NAMESPACE        NAME                                   CLUSTER   APPROVED   ROLE
STAGE
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0411   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0412   hosted    true       auto-assign
clusters-hosted   aaaaaaaa-aaaa-aaaa-aaaa-aaaaaaaa0413   hosted    true       auto-assign
```

3. また、ノードプールレプリカが設定されていることにも注意してください。

```
NAMESPACE   NAME     CLUSTER   DESIRED NODES   CURRENT NODES
AUTOSCALING   AUTOREPAIR   VERSION                    UPDATINGVERSION
UPDATINGCONFIG   MESSAGE
clusters    hosted   hosted    3                       False        False        4.x.y-x86_64
Minimum availability requires 3 replicas, current 0 available
```

4.x.y を、使用するサポートされている OpenShift Container Platform バージョンに置き換えます。

4. ノードがクラスターに参加するまで待ちます。プロセス中に、エージェントはステージとステータスに関する最新情報を提供します。

次に、ホステッドクラスターのデプロイメントを監視します。

1.7.12.7.10. デュアルスタックネットワークのホステッドクラスターデプロイメントの終了

ホステッドクラスターのデプロイメントは、コントロールプレーンとデータプレーンの 2 つの観点から監視できます。

1.7.12.7.10.1. コントロールプレーンを使用してホストされたクラスターのデプロイメントを監視する

コントロールプレーンを使用して、ホストされたクラスターのデプロイメントを監視できます。

1. 次のコマンドを入力して、ホストされたクラスターの **kubeconfig** ファイルをエクスポートします。

```
export KUBECONFIG=<path_to_hosted_cluster_kubeconfig>
```

2. 次のコマンドを入力して、ホストされたクラスターのデプロイメントの進行状況を確認します。

```
watch "oc get pod -n hypershift;echo;echo;oc get pod -n
<hosted_control_plane_namespace>;echo;echo;oc get bmh -A;echo;echo;oc get agent -
A;echo;echo;oc get infraenv -A;echo;echo;oc get hostedcluster -A;echo;echo;oc get
nodepool -A;echo;echo;"
```

このコマンドは、次のアーティファクトに関する情報を提供します。

- HyperShift Operator
- **HostedControlPlane** Pod
- ベアメタルホスト
- エージェント
- **InfraEnv** リソース
- **HostedCluster** および **NodePool** リソース

1.7.12.7.10.2. データプレーンを使用してホストされたクラスターのデプロイメントを監視する

データプレーンを使用して、ホストされたクラスターのデプロイメントプロセス中に Operator がどのように進行しているかを監視できます。

1. 次のコマンドを入力して、ホストされたクラスターの **kubeconfig** ファイルを作成します。

```
hcp create kubeconfig --name <hosted_cluster_name> --namespace
<hosted_cluster_namespace>
```

2. 次のコマンドを入力して、ホストされたクラスターの **kubeconfig** ファイルをエクスポートします。

```
export KUBECONFIG=<path_to_hosted_cluster_kubeconfig>
```

3. 次のコマンドを入力して、クラスターバージョン、ノード、およびクラスター Operator のステータスを確認します。

```
watch "oc get clusterversion,nodes,co"
```

1.7.13. Hosted control plane クラスターの手動インポート

ホステッドクラスターは、Hosted control plane が使用可能になった後、multicluster engine Operator に自動的にインポートされます。ホステッドクラスターを手動でインポートする場合は、次の手順を実行します。

1. **Infrastructure** > **Clusters** をクリックし、インポートするホステッドクラスターを選択します。

2. **Import hosted cluster** をクリックします。
注記: 検出された ホステッドクラスターについては、コンソールからインポートすることもできますが、クラスターはアップグレード可能な状態である必要があります。Hosted control plane を使用できないため、ホステッドクラスターがアップグレード可能な状態ではない場

合、クラスターへのインポートは無効になります。**Import** をクリックして、プロセスを開始します。クラスターが更新を受信している間、ステータスは **Importing** であり、その後、**Ready** に変わります。

1.7.13.1. Hosted control plane クラスターの AWS での手動インポート

次の手順を実行することで、コマンドラインインターフェイスを使用して、ホストされているコントロールプレーンクラスターを AWS にインポートすることもできます。

1. 以下のサンプル YAML ファイルを使用して、**ManagedCluster** リソースを作成します。

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  annotations:
    import.open-cluster-management.io/hosting-cluster-name: local-cluster
    import.open-cluster-management.io/klusterlet-deploy-mode: Hosted
    open-cluster-management/created-via: hypershift
  labels:
    cloud: auto-detect
    cluster.open-cluster-management.io/clusterset: default
    name: <cluster_name>
    vendor: OpenShift
  name: <cluster_name>
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

<cluster_name> は、ホステッドクラスターの名前に置き換えます。

2. 以下のコマンドを実行してリソースを適用します。

```
oc apply -f <file_name>
```

<file_name> を、直前の手順で作成した YAML ファイル名に置き換えます。

3. 以下のサンプル YAML ファイルを使用して、**KlusterletAddonConfig** リソースを作成します。これは、Red Hat Advanced Cluster Management にのみ適用されます。multicluster engine Operator のみをインストールした場合は、この手順を省略します。

```
apiVersion: agent.open-cluster-management.io/v1
kind: KlusterletAddonConfig
metadata:
  name: <cluster_name>
  namespace: <cluster_name>
spec:
  clusterName: <cluster_name>
  clusterNamespace: <cluster_name>
  clusterLabels:
    cloud: auto-detect
    vendor: auto-detect
  applicationManager:
    enabled: true
  certPolicyController:
    enabled: true
```

```
  iamPolicyController:
    enabled: true
  policyController:
    enabled: true
  searchCollector:
    enabled: false
```

<cluster_name> は、ホステッドクラスターの名前に置き換えます。

4. 以下のコマンドを実行してリソースを適用します。

```
oc apply -f <file_name>
```

<file_name> を、直前の手順で作成した YAML ファイル名に置き換えます。

5. インポートプロセスが完了すると、ホステッドクラスターがコンソールに表示されます。以下のコマンドを実行して、ホステッドクラスターのステータスを確認することもできます。

```
oc get managedcluster <cluster_name>
```

1.7.13.2. 関連情報

- ホステッドクラスターの自動インポートを無効にする手順は、multicluster engine operator へのホステッドクラスターの自動インポートの無効化 を参照してください。

1.7.13.3. ホステッドクラスターの Multi-Cluster Engine Operator への自動インポートを無効にする

コントロールプレーンが使用可能になった後、ホステッドクラスターが Multi-Cluster Engine Operator に自動的にインポートされます。

自動インポートを無効にしても、以前にインポートされたホステッドクラスターは影響を受けません。Multi-Cluster Engine Operator 2.5 にアップグレードし、自動インポートが有効になっている場合、コントロールプレーンが使用可能な場合、インポートされていないすべてのホストクラスターが自動的にインポートされます。

注記: Red Hat Advanced Cluster Management がインストールされている場合は、すべての Red Hat Advanced Cluster Management アドオンも有効になります。

自動インポートが無効になっている場合、新しく作成されたホステッドクラスターのみが自動的にインポートされません。すでにインポートされているホステッドクラスターは影響を受けません。コンソールを使用するか、**ManagedCluster** および **KlusterletAddonConfig** カスタムリソースを作成することにより、クラスターを手動でインポートすることもできます。

ホステッドクラスターの自動インポートを無効にするには、次の手順を実行します。

1. マルチクラスターエンジン Operator ハブクラスターで、**AddonDeploymentConfig** リソースを開き、**multicluster-engine** namespace の **hypershift-addon-deploy-config** 仕様を編集します。以下のコマンドを入力します。

```
oc edit addondeploymentconfig hypershift-addon-deploy-config -n multicluster-engine
```

2. 次の例に示すように、**spec.customizedVariables** セクションで、値が **"true"** の **autoImportDisabled** 変数を追加します。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: AddOnDeploymentConfig
metadata:
  name: hypershift-addon-deploy-config
  namespace: multicluster-engine
spec:
  customizedVariables:
  - name: autoImportDisabled
    value: "true"
```

3. 自動インポートを再度有効にするには、**autoImportDisabled** 変数の値を **false** に設定するか、**AddonDeploymentConfig** リソースから変数を削除します。

1.7.13.3.1. 関連情報

ホステッドクラスターを手動でインポートする手順は、Hosted control plane クラスターの手動インポート を参照してください。

1.7.14. Hosted control plane 機能の有効化または無効化

Hosted control plane 機能と **hypershift-addon** マネージドクラスターアドオンは、デフォルトで有効になっています。機能を無効にする場合、または機能を無効にして手動で有効にする必要がある場合は、次の手順を参照してください。

- Hosted control plane 機能を手動での有効化
- Hosted control plane 機能の無効化

1.7.14.1. Hosted control plane 機能を手動での有効化

1. 次のコマンドを実行して、以下の機能を有効にすることができます。

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift","enabled": true}]}}}' 1
```

1 デフォルトの **MultiClusterEngine** リソースインスタンス名は **multiclusterengine** ですが、**$ oc get mce** コマンドを実行し、クラスターから **MultiClusterEngine** 名を取得できます。

2. 次のコマンドを実行して、**hypershift** および **hypershift-local-hosting** 機能が **MultiClusterEngine** カスタムリソースで有効になっていることを確認します。

```
oc get mce multiclusterengine -o yaml 1
```

1 デフォルトの **MultiClusterEngine** リソースインスタンス名は **multiclusterengine** ですが、**$ oc get mce** コマンドを実行し、クラスターから **MultiClusterEngine** 名を取得できます。

出力は以下の例のようになります。

```
apiVersion: multicluster.openshift.io/v1
kind: MultiClusterEngine
```

```
metadata:
  name: multiclusterengine
spec:
  overrides:
    components:
    - name: hypershift
      enabled: true
    - name: hypershift-local-hosting
      enabled: true
```

1.7.14.1.1. local-cluster の hypershift-addon マネージドクラスターアドオンを手動で有効にする

Hosted control plane 機能を有効にすると、**hypershift-addon** マネージドクラスターアドオンが自動的に有効になります。**hypershift-addon** マネージドクラスターアドオンを手動で有効にする必要がある場合は、次の手順を実行して **hypershift-addon** を使用し、HyperShift Operator を **local-cluster** にインストールします。

1. 以下の例のようなファイルを作成して、**ManagedClusterAddon** HyperShift アドオンを作成します。

```
apiVersion: addon.open-cluster-management.io/v1alpha1
kind: ManagedClusterAddOn
metadata:
  name: hypershift-addon
  namespace: local-cluster
spec:
  installNamespace: open-cluster-management-agent-addon
```

2. 以下のコマンドを実行してこのファイルを適用します。

```
oc apply -f <filename>
```

filename は、作成したファイル名に置き換えます。

3. 以下のコマンドを実行して、**hypershift-addon** がインストールされていることを確認します。

```
oc get managedclusteraddons -n local-cluster hypershift-addon
```

アドオンがインストールされている場合、出力は以下の例のようになります。

```
NAME              AVAILABLE   DEGRADED   PROGRESSING
hypershift-addon   True
```

HyperShift アドオンがインストールされ、ホスティングクラスターを使用してホステッドクラスターを作成および管理できるようになります。

1.7.14.2. Hosted control plane 機能の無効化

HyperShift Operator をアンインストールして、Hosted control plane を無効にすることができます。Hosted control plane クラスター機能を無効にする場合は、 **Hosted control plane クラスターの管理** トピックで説明されているとおり、multicluster engine operator でホステッドクラスターとマネージドクラスターリソースを破棄する必要があります。

1.7.14.2.1. HyperShift Operator のアンインストール

HyperShift Operator をアンインストールし、**local-cluster** から **hypershift-addon** を無効にするには、以下の手順を実行します。

1. 以下のコマンドを実行して、ホステッドクラスターが実行されていないことを確認します。

```
oc get hostedcluster -A
```

重要: ホステッドクラスターが実行されている場合、**hypershift-addon** が無効になっていても、HyperShift Operator はアンインストールされません。

2. 以下のコマンドを実行して **hypershift-addon** を無効にします。

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift-local-hosting","enabled": false}]}}}' 1
```

1 デフォルトの **MultiClusterEngine** リソースインスタンス名は **multiclusterengine** ですが、**$ oc get mce** コマンドを実行し、クラスターから **MultiClusterEngine** 名を取得できます。

注記: hypershift-addon を無効にした後、multicluster engine Operator コンソールから **local-cluster** の **hypershift-addon** を無効にすることもできます。

1.7.14.2.2. Hosted control plane 機能の無効化

Hosted control plane 機能を無効にする前に、まず HyperShift Operator をアンインストールする必要があります。次のコマンドを実行して、Hosted control plane 機能を無効にします。

```
oc patch mce multiclusterengine --type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift","enabled": false}]}}}' 1
```

1 デフォルトの **MultiClusterEngine** リソースインスタンス名は **multiclusterengine** ですが、**$ oc get mce** コマンドを実行し、クラスターから **MultiClusterEngine** 名を取得できます。

次のコマンドを実行すると、**MultiClusterEngine** カスタムリソースで **hypershift** および **hypershift-local-hosting** 機能が無効になっていることを確認できます。

```
oc get mce multiclusterengine -o yaml 1
```

1 デフォルトの **MultiClusterEngine** リソースインスタンス名は **multiclusterengine** ですが、**$ oc get mce** コマンドを実行し、クラスターから **MultiClusterEngine** 名を取得できます。

hypershift-preview と **hypershift-local-hosting** の **enabled:** フラグが **false** に設定されている次の例を参照してください。

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

```
components:
- name: hypershift
  enabled: false
- name: hypershift-local-hosting
  enabled: false
```

1.7.14.3. 関連情報

- AWS での Hosted Control Plane クラスターの設定 (テクノロジープレビュー)
- ベアメタルでの Hosted control plane クラスターの設定

1.8. API

multicluster engine operator を使用して、クラスターのライフサイクル管理のために次の API にアクセスできます。**ユーザーに必要なアクセス権限:** 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。

注:統合コンソールからすべての API にアクセスすることもできます。**local-cluster** ビューから、**Home** > **API Explorer** に移動して、API グループを確認します。

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 (v1beta2)
- Clusterview API
- ClusterSetBindings API (v1beta2)
- MultiClusterEngine API
- Placements API (v1beta1)
- PlacementDecisions API (v1beta1)
- ManagedServiceAccount API

1.8.1. Clusters API

1.8.1.1. 概要

このドキュメントは、Kubernetes Operator 用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。

1.8.1.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.1.1.2. タグ

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

1.8.1.2. パス

1.8.1.2.1. 全クラスターのクエリー

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

1.8.1.2.1.1. 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。

1.8.1.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.1.2.1.3. 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

1.8.1.2.1.4. 消費

- **cluster/yaml**

1.8.1.2.1.5. タグ

- cluster.open-cluster-management.io

1.8.1.2.2. クラスターの作成

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

1.8.1.2.2.1. 説明

クラスターの作成

1.8.1.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**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

1.8.1.2.2.3. 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

1.8.1.2.2.4. 消費

- **cluster/yaml**

1.8.1.2.2.5. タグ

- cluster.open-cluster-management.io

1.8.1.2.2.6. HTTP リクエストの例

1.8.1.2.2.6.1. 要求の body

```
{
  "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" : { }
}
```

1.8.1.2.3. 単一クラスターのクエリー

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

1.8.1.2.3.1. 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。

1.8.1.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**cluster_name** **必須**	問い合わせるクラスターの名前。	string

1.8.1.2.3.3. 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

1.8.1.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.1.2.4. クラスターの削除

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

1.8.1.2.4.1. 説明

単一クラスターを削除します。

1.8.1.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**cluster_name** **必須**	削除するクラスターの名前。	string

1.8.1.2.4.3. 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

1.8.1.2.4.4. タグ

- cluster.open-cluster-management.io

1.8.1.3. 定義

1.8.1.3.1. クラスター

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

spec

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

managedClusterClientConfigs

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

1.8.2. Clustersets API (v1beta2)

1.8.2.1. 概要

このドキュメントは、Kubernetes Operator 用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。

1.8.2.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.2.1.2. タグ

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

1.8.2.2. パス

1.8.2.2.1. 全 clusterset のクエリー

```
GET /cluster.open-cluster-management.io/v1beta2/managedclustersets
```

1.8.2.2.1.1. 説明

Clustersets に対してクエリーを実行して詳細を確認します。

1.8.2.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.2.2.1.3. 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

1.8.2.2.1.4. 消費

- **clusterset/yaml**

1.8.2.2.1.5. タグ

- cluster.open-cluster-management.io

1.8.2.2.2. clusterset の作成

```
POST /cluster.open-cluster-management.io/v1beta2/managedclustersets
```

1.8.2.2.2.1. 説明

Clusterset を作成します。

1.8.2.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**body** **必須**	作成する clusterset を記述するパラメーター	Clusterset

1.8.2.2.2.3. 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

1.8.2.2.2.4. 消費

- **clusterset/yaml**

1.8.2.2.2.5. タグ

- cluster.open-cluster-management.io

1.8.2.2.2.6. HTTP リクエストの例

1.8.2.2.2.6.1. 要求の body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta2",
  "kind" : "ManagedClusterSet",
  "metadata" : {
    "name" : "clusterset1"
  },
  "spec": { },
  "status" : { }
}
```

1.8.2.2.3. 単一 clusterset のクエリー

```
GET /cluster.open-cluster-management.io/v1beta2/managedclustersets/{clusterset_name}
```

1.8.2.2.3.1. 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。

1.8.2.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterset_name** **必須**	問い合わせる clusterset の名前。	string

1.8.2.2.3.3. 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

1.8.2.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.2.2.4. clusterset の削除

```
DELETE /cluster.open-cluster-management.io/v1beta2/managedclustersets/{clusterset_name}
```

1.8.2.2.4.1. 説明

単一 clusterset を削除します。

1.8.2.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterset_name** **必須**	削除する clusterset の名前。	string

1.8.2.2.4.3. 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

1.8.2.2.4.4. タグ

- cluster.open-cluster-management.io

1.8.2.3. 定義

1.8.2.3.1. Clusterset

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

1.8.3. Clustersetbindings API (v1beta2)

1.8.3.1. 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 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。

1.8.3.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.3.1.2. タグ

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

1.8.3.2. パス

1.8.3.2.1. 全 clustersetbinding のクエリー

```
GET /cluster.open-cluster-
management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings
```

1.8.3.2.1.1. 説明

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

1.8.3.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**namespace** **必須**	使用する namespace (例: default)	string

型	名前	説明	スキーマ

1.8.3.2.1.3. 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

1.8.3.2.1.4. 消費

- **clustersetbinding/yaml**

1.8.3.2.1.5. タグ

- cluster.open-cluster-management.io

1.8.3.2.2. clustersetbinding の作成

> POST /cluster.open-cluster-management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings

1.8.3.2.2.1. 説明

clustersetbinding を作成します。

1.8.3.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**namespace 必須**	使用する namespace (例: default)	string

型	名前	説明	スキーマ
Body	body 必須	作成する clustersetbinding を記述するパラメーター	Clustersetbinding

1.8.3.2.2.3. 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

1.8.3.2.2.4. 消費

- **clustersetbinding/yaml**

1.8.3.2.2.5. タグ

- cluster.open-cluster-management.io

1.8.3.2.2.6. HTTP リクエストの例

1.8.3.2.2.6.1. 要求の body

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

1.8.3.2.3. 単一 clustersetbinding のクエリー

```
GET /cluster.open-cluster-management.io/v1beta2/namespaces/{namespace}/managedclustersetbindings/{clustersetbinding_name}
```

1.8.3.2.3.1. 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。

1.8.3.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**namespace** **必須**	使用する namespace (例: default)	string
パス	**clustersetbinding_name** **必須**	問い合わせる clustersetbinding の名前	string

1.8.3.2.3.3. 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

1.8.3.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.3.2.4. clustersetbinding の削除

```
DELETE /cluster.open-cluster-management.io/v1beta2/managedclustersetbindings/{clustersetbinding_name}
```

1.8.3.2.4.1. 説明

単一 clustersetbinding を削除します。

1.8.3.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**namespace** **必須**	使用する namespace (例: default)	string
パス	**clustersetbinding_name** **必須**	削除する clustersetbinding の名前	string

1.8.3.2.4.3. 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

1.8.3.2.4.4. タグ

- cluster.open-cluster-management.io

1.8.3.3. 定義

1.8.3.3.1. Clustersetbinding

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

名前	スキーマ
metadata **必須**	object
spec **必須**	spec

spec

名前	スキーマ
clusterSet **必須**	string

1.8.4. Clusterview API (v1alpha1)

1.8.4.1. 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 **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リストを表示できる 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。

1.8.4.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.4.1.2. 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リストを表示します。

1.8.4.2. パス

1.8.4.2.1. マネージドクラスターの取得

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

1.8.4.2.1.1. 説明

アクセス可能なマネージドクラスターのリストを表示します。

1.8.4.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.4.2.1.3. 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

1.8.4.2.1.4. 消費

- **managedcluster/yaml**

1.8.4.2.1.5. タグ

- clusterview.open-cluster-management.io

1.8.4.2.2. マネージドクラスターのリスト表示

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

1.8.4.2.2.1. 説明

アクセス可能なマネージドクラスターのリストを表示します。

1.8.4.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**body** **任意**	マネージドクラスターをリスト表示するユーザー ID の名前	string

1.8.4.2.2.3. 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

1.8.4.2.2.4. 消費

- **managedcluster/yaml**

1.8.4.2.2.5. タグ

- clusterview.open-cluster-management.io

1.8.4.2.2.6. HTTP リクエストの例

1.8.4.2.2.6.1. 要求の body

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

1.8.4.2.3. マネージドクラスターセットの監視

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

1.8.4.2.3.1. 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。

1.8.4.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	string

1.8.4.2.3.3. 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

1.8.4.2.4. マネージドクラスターセットのリスト表示

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

1.8.4.2.4.1. 説明

アクセス可能なマネージドクラスターをリスト表示します。

1.8.4.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	string

1.8.4.2.4.3. 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

1.8.4.2.5. マネージドクラスターセットのリスト表示

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

1.8.4.2.5.1. 説明

アクセス可能なマネージドクラスターをリスト表示します。

1.8.4.2.5.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	string

1.8.4.2.5.3. 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

1.8.4.2.6. マネージドクラスターセットの監視

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

1.8.4.2.6.1. 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。

1.8.4.2.6.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**clusterview_name** **任意**	監視するユーザー ID の名前	string

1.8.4.2.6.3. 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

1.8.5. ManagedServiceAccount API (v1alpha1) (非推奨)

1.8.5.1. 概要

このドキュメントは、multicluster engine operator 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。

非推奨: v1alpha1 API は非推奨になりました。最適な結果を得るには、代わりに **v1beta1** を使用します。

1.8.5.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.5.1.2. タグ

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

1.8.5.2. パス

1.8.5.2.1. ManagedServiceAccount を作成する

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

1.8.5.2.1.1. 説明

ManagedServiceAccount を作成します。

1.8.5.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**body** **必須**	作成する ManagedServiceAccount を説明するパラメーター。	ManagedServiceAccount

1.8.5.2.1.3. 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

1.8.5.2.1.4. 消費

- **managedserviceaccount/yaml**

1.8.5.2.1.5. タグ

- managedserviceaccount.multicluster.openshift.io

1.8.5.2.1.5.1. 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": "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": []
}
}
```

1.8.5.2.2. 単一の ManagedServiceAccount をクエリーする

> GET /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccount_name}

1.8.5.2.2.1. 説明

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

1.8.5.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	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。	string
パス	managedserviceaccount_name **必須**	照会する **ManagedServiceAccount** の名前。	string

1.8.5.2.2.3. 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

1.8.5.2.2.4. タグ

- cluster.open-cluster-management.io

1.8.5.2.3. ManagedServiceAccount を削除する

> DELETE /cluster.open-cluster-management.io/v1alpha1/namespaces/{namespace}/managedserviceaccounts/{managedserviceaccount_name}

1.8.5.2.3.1. 説明

単一の **ManagedServiceAccount** を削除します。

1.8.5.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**managedserviceaccount_name** **必須**	削除する **ManagedServiceAccount** の名前。	string

1.8.5.2.3.3. 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

1.8.5.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.5.3. 定義

1.8.5.3.1. 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

名前	説明	スキーマ
metadata **必須**	**ManagedServiceAccount** のメタデータ。	object
spec **必須**	**ManagedServiceAccount** の仕様。	

1.8.6. MultiClusterEngine API (v1alpha1)

1.8.6.1. 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。

1.8.6.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.6.1.2. タグ

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

1.8.6.2. パス

1.8.6.2.1. MultiClusterEngine を作成する

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

1.8.6.2.1.1. 説明

MultiClusterEngine を作成します。

1.8.6.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	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。	string
Body	body **必須**	作成する MultiClusterEngine を説明するパラメーター。	MultiClusterEngine

1.8.6.2.1.3. 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

1.8.6.2.1.4. 消費

- **MultiClusterEngines/yaml**

1.8.6.2.1.5. タグ

- multiclusterengines.multicluster.openshift.io

1.8.6.2.1.5.1. 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": []
 }
}
```


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

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

1.8.6.2.2.1. 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。

1.8.6.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.6.2.2.3. 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

1.8.6.2.2.4. 消費

- **operator/yaml**

1.8.6.2.2.5. タグ

- multiclusterengines.multicluster.openshift.io

1.8.6.2.3. MultiClusterEngine Operator の削除

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

1.8.6.2.3.1. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Path	**name** **必須**	削除するマルチクラスターエンジンの名前。	string

1.8.6.2.3.2. 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

HTTP コード	説明	スキーマ
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

1.8.6.2.3.3. タグ

- multiclusterengines.multicluster.openshift.io

1.8.6.3. 定義

1.8.6.3.1. 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。	**仕様のリスト** を参照してください。

1.8.6.3.2. 仕様のリスト

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

名前	説明	スキーマ
targetNamespace **任意**	MCE リソースが配置される場所。	string

1.8.7. Placements API (v1beta1)

1.8.7.1. 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。

1.8.7.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.7.1.2. タグ

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

1.8.7.2. パス

1.8.7.2.1. 全 Placement のクエリー

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

1.8.7.2.1.1. 説明

Placement に対してクエリーを実行して詳細を確認します。

1.8.7.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.7.2.1.3. 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

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

1.8.7.2.1.4. 消費

- **placement/yaml**

1.8.7.2.1.5. タグ

- cluster.open-cluster-management.io

1.8.7.2.2. Placement の作成

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

1.8.7.2.2.1. 説明

Placement を作成します。

1.8.7.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**body** **必須**	作成する placement を記述するパラメーター	Placement

1.8.7.2.2.3. 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

HTTP コード	説明	スキーマ
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

1.8.7.2.2.4. 消費

- **placement/yaml**

1.8.7.2.2.5. タグ

- cluster.open-cluster-management.io

1.8.7.2.2.6. HTTP リクエストの例

1.8.7.2.2.6.1. 要求の body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "Placement",
  "metadata" : {
    "name" : "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status" : { }
}
```

1.8.7.2.3. 単一の Placement のクエリー

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

1.8.7.2.3.1. 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。

1.8.7.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	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。	string
パス	placement_name 必須	問い合わせる Placement の名前	string

1.8.7.2.3.3. 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

1.8.7.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.7.2.4. Placement の削除

```
DELETE /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placements/{placement_name}
```

1.8.7.2.4.1. 説明

単一の Placement を削除します。

1.8.7.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	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。	string

型	名前	説明	スキーマ
パス	**placement_name** **必須**	削除する Placement の名前	string

1.8.7.2.4.3. 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

1.8.7.2.4.4. タグ

- cluster.open-cluster-management.io

1.8.7.3. 定義

1.8.7.3.1. 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。	文字列配列
numberOfClusters **任意**	選択する ManagedClusters の必要数	integer (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
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

1.8.8. PlacementDecisions API (v1beta1)

1.8.8.1. 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。

1.8.8.1.1. URI スキーム

ベースパス: /kubernetes/apis
スキーム: HTTPS

1.8.8.1.2. タグ

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

1.8.8.2. パス

1.8.8.2.1. 全 PlacementDecision のクエリー

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

1.8.8.2.1.1. 説明

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

1.8.8.2.1.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string

1.8.8.2.1.3. 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

1.8.8.2.1.4. 消費

- **placementdecision/yaml**

1.8.8.2.1.5. タグ

- cluster.open-cluster-management.io

1.8.8.2.2. PlacementDecision の作成

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

1.8.8.2.2.1. 説明

PlacementDecisions を作成します。

1.8.8.2.2.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
Body	**body** **必須**	作成する PlacementDecision を記述するパラメーター	PlacementDecision

1.8.8.2.2.3. 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

1.8.8.2.2.4. 消費

- **placementdecision/yaml**

1.8.8.2.2.5. タグ

- cluster.open-cluster-management.io

1.8.8.2.2.6. HTTP リクエストの例

1.8.8.2.2.6.1. 要求の body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "PlacementDecision",
  "metadata" : {
    "labels" : {
      "cluster.open-cluster-management.io/placement" : "placement1"
    },
    "name" : "placement1-decision1",
    "namespace": "ns1"
  },
  "status" : { }
}
```

1.8.8.2.3. 単一の PlacementDecision のクエリー

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

1.8.8.2.3.1. 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。

1.8.8.2.3.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**placementdecision_name** **必須**	問い合わせる PlacementDecision の名前	string

1.8.8.2.3.3. 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

1.8.8.2.3.4. タグ

- cluster.open-cluster-management.io

1.8.8.2.4. PlacementDecision の削除

```
DELETE /cluster.open-cluster-
management.io/v1beta1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

1.8.8.2.4.1. 説明

単一の PlacementDecision を削除します。

1.8.8.2.4.2. パラメーター

型	名前	説明	スキーマ
Header	**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。	string
パス	**placementdecision_name** **必須**	削除する PlacementDecision の名前	string

1.8.8.2.4.3. 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

1.8.8.2.4.4. タグ

- cluster.open-cluster-management.io

1.8.8.3. 定義

1.8.8.3.1. 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	object

1.9. トラブルシューティング

トラブルシューティングガイドをご使用の前に **oc adm must-gather** コマンドを実行して、詳細およびログを収集し、問題のデバッグ手順を行います。詳細は、must-gather コマンドを実行したトラブルシューティング を参照してください。

また、ロールベースのアクセス権限を確認してください。詳細は、multicluster engine operator のロールベースのアクセス制御 を参照してください。

1.9.1. 文書化されたトラブルシューティング

multicluster engine operator のトラブルシューティングトピックのリストを表示します。

インストール:

インストールタスクに関する主要なドキュメントを表示するには、マルチクラスターエンジンオペレータのインストールとアップグレード を参照してください。

- インストールステータスがインストールまたは保留中の状態のトラブルシューティング
- 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

クラスター管理:

クラスターの管理に関する主要なドキュメントを表示するには、クラスターライフサイクルの概要 を参照してください。

- 既存のクラスターに Day-2 ノードを追加するトラブルシューティングが保留中のユーザー操作で失敗する
- オフラインクラスターのトラブルシューティング
- マネージドクラスターのインポート失敗に関するトラブルシューティング
- クラスターの再インポートが不明な権限エラーで失敗する
- Pending Import ステータスのクラスターのトラブルシューティング
- 証明書を変更した後のインポート済みクラスターのオフラインでのトラブルシューティング
- クラスターのステータスが offline から available に変わる場合のトラブルシューティング
- VMware vSphere でのクラスター作成のトラブルシューティング
- ステータスが Pending または Failed のクラスターのコンソールでのトラブルシューティング
- OpenShift Container Platform バージョン 3.11 クラスターのインポートの失敗時のトラブルシューティング
- degraded 状態にある Klusterlet のトラブルシューティング
- クラスターの削除後も namespace が残る
- クラスターのインポート時の auto-import-secret-exists エラー
- Troubleshooting missing PlacementDecision after creating Placement
- Dell ハードウェアにおけるベアメタルホストの検出エラーのトラブルシューティング
- 最小限の ISO 起動エラーのトラブルシューティング

1.9.2. 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

1.9.2.1. 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。

1.9.2.2. 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. OpenShift Container Platform ドキュメントの クラスターに関するデータの収集 で、**must-gather** コマンドについて確認し、必要なもの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ます。

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel9:v2.5 --dest-dir=<directory>
```

1. 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: クラスタースコープおよび 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

1.9.2.3. 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。**sha256** はｍ現在のイメージに置き換えます。

```
REGISTRY=registry.example.com:5000
IMAGE=$REGISTRY/multicluster-engine/must-gather-rhel9@sha256:ff9f37eb400dc1f7d07a9b6f2da9064992934b69847d17f59e385783c071b9d8>

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

ここ で製品チーム向けの Jira バグを作成できます。

- must-gather コマンドを実行したトラブルシューティング

1.9.2.4. ホステッドクラスターの Must-gather

Hosted control plane クラスターで問題が発生した場合は、**must-gather** コマンドを実行して、トラブルシューティングに役立つ情報を収集できます。

1.9.2.4.1. ホステッドクラスターの must-gather コマンドについて

このコマンドは、管理クラスターとホストされたクラスターの出力を生成します。

- multicluster engine Operator ハブクラスターからのデータ:
  - クラススコープのリソース: これらのリソースは、管理クラスターのノード定義です。
  - **hypershift-dump** 圧縮ファイル: このファイルは、コンテンツを他の人と共有する必要がある場合に役立ちます。
  - namespace リソース: これらのリソースには、config map、サービス、イベント、ログなど、関連する namespace のすべてのオブジェクトが含まれます。
  - ネットワークログ: これらのログには、OVN ノースバウンドデータベースとサウスバウンドデータベース、およびそれぞれのステータスが含まれます。
  - ホストされたクラスター: このレベルの出力には、ホストされたクラスター内のすべてのリソースが含まれます。
- ホストされたクラスターからのデータ:
  - クラススコープのリソース: これらのリソースには、ノードや CRD などのクラスター全体のオブジェクトがすべて含まれます。
  - namespace リソース: これらのリソースには、config map、サービス、イベント、ログなど、関連する namespace のすべてのオブジェクトが含まれます。

出力にはクラスターからのシークレットオブジェクトは含まれませんが、シークレットの名前への参照が含まれる可能性があります。

1.9.2.4.2. 前提条件

must-gather コマンドを実行して情報を収集するには、次の前提条件を満たす必要があります。

- **kubeconfig** ファイルが読み込まれ、multicluster engine Operator ハブクラスターを指している。
- multicluster engine Operator ハブクラスターへの cluster-admin アクセスがある。
- **HostedCluster** リソースの name 値と、カスタムリソースがデプロイされる namespace がある。

1.9.2.4.3. ホスト型クラスターの must-gather コマンドの入力

1. 以下のコマンドを実行して、ホスト型クラスターに関する情報を収集します。このコマンドでは、**hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE** パラメーターはオプションです。これを含めない場合、コマンドは、ホストされたクラスターがデフォルトの namespace (**clusters**) 内にあるかのように実行されます。

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.x
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME
```

2. コマンドの結果を圧縮ファイルに保存するには、**--dest-dir=NAME** パラメーターを含めて、**NAME** は、結果を保存するディレクトリーの名前に置き換えます。

```
oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.x
/usr/bin/gather hosted-cluster-namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME --dest-dir=NAME ; tar -cvzf NAME.tgz NAME
```

1.9.2.4.4. 非接続環境での must-gather コマンドの入力

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 /usr/bin/gather hosted-cluster-
namespace=HOSTEDCLUSTERNAMESPACE hosted-cluster-
name=HOSTEDCLUSTERNAME --dest-dir=./data
```

1.9.2.4.5. 関連情報

- Hosted control plane のトラブルシューティングの詳細は、OpenShift Container Platform ドキュメントの Hosted control plane のトラブルシューティング を参照してください。

1.9.3. 既存のクラスターに Day-2 ノードを追加するトラブルシューティングが保留中のユーザー操作で失敗する

インストール時に、ゼロタッチプロビジョニングまたはホストのインベントリー作成メソッドで Kubernetes Operator のマルチクラスターエンジンが作成した既存のクラスターにノードを追加したり、スケールアウトできません。インストールプロセスは、検出フェーズでは正しく機能しますが、インストールフェーズでは失敗します。

ネットワークの設定に失敗しています。統合コンソールのハブクラスターから、**Pending** のユーザーアクションが表示されます。説明から、再起動ステップで失敗していることがわかります。

インストールするホストで実行されているエージェントは情報を報告できないため、失敗に関するエラーメッセージはあまり正確ではありません。

1.9.3.1. 現象: Day 2 ワーカーのインストールが失敗する

検出フェーズの後、ホストは再起動してインストールを続行しますが、ネットワークを設定できません。以下の現象およびメッセージを確認します。

- 統合コンソールのハブクラスターから、追加ノード上で **Pending** ユーザーアクションがないか、**Rebooting** インジケーターが付いているかどうかを確認します。

```
This host is pending user action. Host timed out when pulling ignition. Check the host
console... Rebooting
```

- Red Hat OpenShift Container Platform 設定のマネージドクラスターから、既存のクラスターの **MachineConfig** を確認します。**MachineConfig** のいずれかが次のディレクトリーにファイルを作成しているかどうかを確認します。
  - **/sysroot/etc/NetworkManager/system-connections/**
  - **/sysroot/etc/sysconfig/network-scripts/**
- インストールするホストの端末から、障害が発生したホストに次のメッセージが表示されているかどうかを確認します。**journalctl** を使用してログメッセージを確認できます。

```
info: networking config is defined in the real root

info: will not attempt to propagate initramfs networking
```

ログに最後のメッセージが表示された場合、**現象** に記載されているフォルダーで既存のネットワーク設定がすでに見つかっているため、ネットワーク設定は伝播されません。

1.9.3.2. 問題の解決: ネットワーク設定をマージするノードを再作成します。

インストール中に適切なネットワーク設定を使用するには、次のタスクを実行します。

1. ハブクラスターからノードを削除します。
2. 同じようにノードをインストールするには、前のプロセスを繰り返します。
3. 次のアノテーションを使用してノードの **BareMetalHost** オブジェクトを作成します。

```
"bmac.agent-install.openshift.io/installer-args": "[\"--append-karg\",
\"coreos.force_persist_ip\"]"
```

ノードがインストールを開始します。検出フェーズの後、ノードは既存のクラスター上の変更と初期設定の間でネットワーク設定をマージします。

1.9.4. エージェントプラットフォーム上の Hosted Control Plane クラスター削除失敗のトラブルシューティング

エージェントプラットフォーム上の Hosted Control Plane クラスターを破棄すると、通常、すべてのバックエンドリソースが削除されます。マシンリソースが正しく削除されていない場合、クラスターの削除が失敗します。この場合、残りのマシンリソースを手動で削除する必要があります。

1.9.4.1. 現象: Hosted Control Plane クラスターを破棄するとエラーが発生する

エージェントプラットフォーム上の Hosted Control Plane クラスターを破棄しようとすると、**hcp destroy** コマンドが次のエラーで失敗します。

+

```
2024-02-22T09:56:19-05:00    ERROR    HostedCluster deletion failed    {"namespace": "clusters",
"name": "hosted-0", "error": "context deadline exceeded"}
2024-02-22T09:56:19-05:00    ERROR    Failed to destroy cluster    {"error": "context deadline
```

```
exceeded"}
```

1.9.4.2. 問題の解決方法: 残りのマシンリソースを手動で削除する

エージェントプラットフォーム上の Hosted Control Plane クラスターを正常に破棄するには、次の手順を実行します。

1. 次のコマンドを実行して、残りのマシンリソースのリストを表示します。**<hosted_cluster_namespace>** は、ホストされたクラスターの namespace の名前に置き換えます。

   ```
   oc get machine -n <hosted_cluster_namespace>
   ```

   以下の出力例を参照してください。

   ```
   NAMESPACE NAME CLUSTER NODENAME PROVIDERID PHASE AGE VERSION
   clusters-hosted-0 hosted-0-9gg8b hosted-0-nhdbp Deleting 10h 4.15.0-rc.8
   ```

2. 次のコマンドを実行して、マシンリソースに割り当てられている **machine.cluster.x-k8s.io** ファイナライザーを削除します。

   ```
   oc edit machines -n <hosted_cluster_namespace>
   ```

3. 次のコマンドを実行して、ターミナルに **No resources found** というメッセージが表示されることを確認します。

   ```
   oc get agentmachine -n <hosted_cluster_namespace>
   ```

4. 次のコマンドを実行して、エージェントプラットフォーム上の Hosted Control Plane クラスターを破棄します。

   ```
   hcp destroy cluster agent --name <cluster_name>
   ```

   <cluster_name> は、クラスター名に置き換えます。

1.9.5. インストールステータスがインストールまたは保留中の状態のトラブルシューティング

multicluster engine operator をインストールするときに、**MultiClusterEngine** が **Installing** フェーズのままであるか、複数の Pod が **Pending** ステータスを維持します。

1.9.5.1. 現象: Pending 状態で止まる

MultiClusterEngine をインストールしてから、**MultiClusterEngine** リソースの **status.components** フィールドからのコンポーネントの 1 つ以上で **ProgressDeadlineExceeded** と報告したまま 10 分以上経過しています。クラスターのリソース制約が問題となっている場合があります。

MultiClusterEngine がインストールされた namespace で Pod を確認します。以下のようなステータスとともに **Pending** と表示される場合があります。

```
reason: Unschedulable
```

```
message: '0/6 nodes are available: 3 Insufficient cpu, 3 node(s) had taint {node-
role.kubernetes.io/master:
        }, that the pod didn't tolerate.'
```

このような場合には、ワーカーノードにはクラスターでの製品実行に十分なリソースがありません。

1.9.5.2. 問題の解決: ワーカーノードのサイズの調整

この問題が発生した場合は、大規模なワーカーノードまたは複数のワーカーノードでクラスターを更新する必要があります。クラスターのサイジングのガイドラインについては、クラスターのサイジング を参照してください。

1.9.6. 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

multicluster engine operator を再インストールすると、Pod が起動しません。

1.9.6.1. 現象: 再インストールの失敗

multicluster engine Operator をインストールした後に Pod が起動しない場合は、multicluster engine Operator の以前のインストールからの項目が、アンインストール時に正しく削除されなかったことが原因であることが多いです。

Pod はこのような場合に、インストールプロセスの完了後に起動しません。

1.9.6.2. 問題の解決: 再インストールの失敗

この問題が発生した場合は、以下の手順を実行します。

1. アンインストール の手順に従い、現在のコンポーネントを削除し、アンインストールプロセスを実行します。
2. Helm のインストール の手順に従い、Helm CLI バイナリーバージョン 3.2.0 以降をインストールします。
3. **oc** コマンドが実行できるように、Red Hat 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の詳細は、OpenShift Container Platform ドキュメントの OpenShift CLI スタートガイド を参照してください。
4. 以下のスクリプトをファイルにコピーします。

```
#!/bin/bash
MCE_NAMESPACE=<namespace>
oc delete multiclusterengine --all
oc delete apiservice v1.admission.cluster.open-cluster-management.io
v1.admission.work.open-cluster-management.io
oc delete crd discoveredclusters.discovery.open-cluster-management.io
discoveryconfigs.discovery.open-cluster-management.io
oc delete mutatingwebhookconfiguration ocm-mutating-webhook
managedclustermutators.admission.cluster.open-cluster-management.io
oc delete validatingwebhookconfiguration ocm-validating-webhook
oc delete ns $MCE_NAMESPACE
```

スクリプト内の **<namespace>** を multicluster engine Operator がインストールされた namespace の名前に置き換えます。namespace が消去され削除されるため、正しい namespace を指定するようにしてください。

5. スクリプトを実行して、アーティファクトを以前のインストールから削除します。
6. インストールを実行します。ネットワーク接続時のオンラインインストール を参照してください。

1.9.7. オフラインクラスターのトラブルシューティング

クラスターのステータスがオフラインと表示される一般的な原因がいくつかあります。

1.9.7.1. 現象: クラスターのステータスがオフライン状態である

クラスターの作成手順を完了したら、Red Hat Advanced Cluster Management コンソールからアクセスできず、クラスターのステータスが **offline** と表示されます。

1.9.7.2. 問題の解決: クラスターのステータスがオフライン状態になっている

1. マネージドクラスターが利用可能かどうかを確認します。これは、Red Hat Advanced Cluster Management コンソールの **Clusters** エリアで確認できます。
利用不可の場合は、マネージドクラスターの再起動を試行します。
2. マネージドクラスターのステータスがオフラインのままの場合は、以下の手順を実行します。
   a. ハブクラスターで **oc get managedcluster <cluster_name> -o yaml** コマンドを実行します。**<cluster_name>** は、クラスター名に置き換えます。
   b. **status.conditions** セクションを見つけます。
   c. **type: ManagedClusterConditionAvailable** のメッセージを確認して、問題を解決します。

1.9.8. マネージドクラスターのインポート失敗に関するトラブルシューティング

クラスターのインポートに失敗した場合は、クラスターのインポートが失敗した理由を判別するためにいくつかの手順を実行できます。

1.9.8.1. 現象: インポートされたクラスターを利用できない

クラスターをインポートする手順を完了すると、コンソールからクラスターにアクセスできなくなります。

1.9.8.2. 問題の解決: インポートされたクラスターが利用できない

インポートの試行後にインポートクラスターが利用できない場合には、いくつかの理由があります。クラスターのインポートに失敗した場合は、インポートに失敗した理由が見つかるまで以下の手順を実行します。

1. ハブクラスターで、次のコマンドを実行して、インポートコントローラーが実行していることを確認します。

```
kubectl -n multicluster-engine get pods -l app=managedcluster-import-controller-v2
```

実行中の Pod が 2 つ表示されるはずです。Pod のいずれかが実行されていない場合には、以下のコマンドを実行してログを表示して理由を判別します。

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1
```

2. ハブクラスターで次のコマンドを実行して、マネージドクラスターのインポートシークレットがインポートコントローラーによって正常に生成されたかどうかを確認します。

```
kubectl -n <managed_cluster_name> get secrets <managed_cluster_name>-import
```

インポートシークレットが存在しない場合は、以下のコマンドを実行してインポートコントローラーのログエントリーを表示し、作成されていない理由を判断します。

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1 | grep importconfig-controller
```

3. ハブクラスターで、マネージドクラスターが **local-cluster** であるか、Hive によってプロビジョニングされているか、自動インポートシークレットがある場合は、次のコマンドを実行して、マネージドクラスターのインポートステータスを確認します。

```
kubectl get managedcluster <managed_cluster_name> -o=jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}{"\t"}{.message}{"\n"}{end}' | grep ManagedClusterImportSucceeded
```

ManagedClusterImportSucceeded が **true** でない場合には、コマンドの結果で失敗の理由が表示されます。

4. マネージドクラスターの Klusterlet ステータスが degraded 状態でないかを確認します。Klusterlet のパフォーマンスが低下した理由を特定するには、degraded 状態にある Klusterlet のトラブルシューティング を参照してください。

1.9.9. クラスターの再インポートが不明な権限エラーで失敗する

マネージドクラスターを multicluster engine Operator に再インポートするときに問題が発生した場合は、手順に従って問題をトラブルシューティングします。

1.9.9.1. 現象: クラスターの再インポートが不明な権限エラーで失敗する

multicluster engine operator を使用して OpenShift Container Platform クラスターをプロビジョニングした後に、API サーバー証明書を変更したり、OpenShift Container Platform クラスターに追加したりすると、**x509: certificate signed by unknown authority** エラーでクラスターの再インポートが失敗する場合があります。

1.9.9.2. 問題の特定: クラスターの再インポートが不明な権限エラーで失敗する

マネージドクラスターの再インポートに失敗した後、次のコマンドを実行して、multicluster engine Operator ハブクラスターのインポートコントローラーログを取得します。

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 -f
```

次のエラーログが表示される場合は、マネージドクラスター API サーバーの証明書が変更されている可能性があります。

ERROR Reconciler error {"controller": "clusterdeployment-controller", "object": {"name":"awscluster1","namespace":"awscluster1"}, "namespace": "awscluster1", "name": "awscluster1", "reconcileID": "a2cccf24-2547-4e26-95fb-f258a6710d80", "error": "Get \"https://api.awscluster1.dev04.red-chesterfield.com:6443/api?timeout=32s\": x509: certificate signed by unknown authority"}

マネージドクラスター API サーバー証明書が変更されたかどうかを確認するには、次の手順を実行します。

1. 次のコマンドを実行して、**your-managed-cluster-name** をマネージドクラスターの名前に置き換えて、マネージドクラスターの名前を指定します。

```
cluster_name=<your-managed-cluster-name>
```

2. 次のコマンドを実行して、マネージドクラスター **kubeconfig** シークレット名を取得します。

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

3. 次のコマンドを実行して、**kubeconfig** を新しいファイルにエクスポートします。

```
oc -n ${cluster_name} get secret ${kubeconfig_secret_name} -ojsonpath={.data.kubeconfig} | base64 -d > kubeconfig.old
```

```
export KUBECONFIG=kubeconfig.old
```

4. 次のコマンドを実行して、**kubeconfig** を使用してマネージドクラスターから namespace を取得します。

```
oc get ns
```

次のメッセージのようなエラーが表示された場合は、クラスター API サーバーの証明書が変更になっており、**kubeconfig** ファイルが無効です。

Unable to connect to the server: x509: certificate signed by unknown authority

1.9.9.3. 問題の解決: クラスターの再インポートが不明な権限エラーで失敗する

マネージドクラスター管理者は、マネージドクラスター用に新しい有効な **kubeconfig** ファイルを作成する必要があります。

新しい **kubeconfig** を作成したら、次の手順を実行して、マネージドクラスターの新しい **kubeconfig** を更新します。

1. 次のコマンドを実行して、**kubeconfig** ファイルパスとクラスター名を設定します。**<path_to_kubeconfig>** を新しい **kubeconfig** ファイルへのパスに置き換えます。**<managed_cluster_name>** をマネージドクラスターの名前に置き換えます。

```
cluster_name=<managed_cluster_name>
kubeconfig_file=<path_to_kubeconfig>
```

2. 次のコマンドを実行して、新しい **kubeconfig** をエンコードします。

```
kubeconfig=$(cat ${kubeconfig_file} | base64 -w0)
```

注記: macOS では、代わりに次のコマンドを実行します。

```
kubeconfig=$(cat ${kubeconfig_file} | base64)
```

3. 次のコマンドを実行して、JSON パッチ **kubeconfig** を定義します。

```
kubeconfig_patch="[{\"op\":\"replace\", \"path\":\"/data/kubeconfig\",
\"value\":\"${kubeconfig}\"}, {\"op\":\"replace\", \"path\":\"/data/raw-kubeconfig\",
\"value\":\"${kubeconfig}\"}]"
```

4. 次のコマンドを実行して、マネージドクラスターから管理者の **kubeconfig** シークレット名を取得します。

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -
ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

5. 次のコマンドを実行して、管理者の **kubeconfig** シークレットに新しい **kubeconfig** を適用します。

```
oc -n ${cluster_name} patch secrets ${kubeconfig_secret_name} --type='json' -
p="${kubeconfig_patch}"
```

1.9.10. Pending Import ステータスのクラスターのトラブルシューティング

クラスターのコンソールで継続的に **Pending import** と表示される場合は、以下の手順を実行して問題をトラブルシューティングしてください。

1.9.10.1. 現象: ステータスが Pending Import クラスター

Red Hat Advanced Cluster Management コンソールを使用してクラスターをインポートした後に、コンソールで、クラスターのステータスが **Pending import** と表示されます。

1.9.10.2. 問題の特定: ステータスが Pending Import クラスター

1. マネージドクラスターで以下のコマンドを実行し、問題のある Kubernetes Pod 名を表示します。

```
kubectl get pod -n open-cluster-management-agent | grep klusterlet-registration-agent
```

2. マネージドクラスターで以下のコマンドを実行し、エラーのログエントリーを探します。

```
kubectl logs <registration_agent_pod> -n open-cluster-management-agent
```

registration_agent_pod は、手順 1 で特定した Pod 名に置き換えます。

3. 返された結果に、ネットワーク接続の問題があったと示すテキストがないかどうかを検索します。たとえば、**no such host** です。

1.9.10.3. 問題の解決: ステータスが Pending Import クラスター

1. ハブクラスターで以下のコマンドを実行して、問題のあるポート番号を取得します。

```
oc get infrastructure cluster -o yaml | grep apiServerURL
```

2. マネージドクラスターのホスト名が解決でき、ホストおよびポートへの送信接続が機能していることを確認します。

マネージドクラスターで通信が確立できない場合は、クラスターのインポートが完了していません。マネージドクラスターのクラスターステータスは、**Pending import** になります。

1.9.11. 証明書を変更した後のインポート済みクラスターのオフラインでのトラブルシューティング

カスタムの **apiserver** 証明書のインストールはサポートされますが、証明書情報を変更する前にインポートされたクラスターの 1 つまたは複数でステータスが **offline** になる可能性があります。

1.9.11.1. 現象: 証明書の変更後にクラスターがオフラインになる

証明書シークレットを更新する手順を完了すると、オンラインだった 1 つ以上のクラスターがコンソールに **offline** ステータスを表示するようになります。

1.9.11.2. 問題の特定: 証明書の変更後にクラスターがオフラインになる

カスタムの API サーバー証明書の情報を更新すると、インポートされ、新しい証明書が追加される前に稼働していたクラスターのステータスが **offline** になります。

オフラインのマネージドクラスターの **open-cluster-management-agent** namespace にある Pod のログで、証明書に問題があるとのエラーが見つかります。以下の例のようなエラーがログに表示されます。

以下の **work-agent** ログを参照してください。

```
E0917 03:04:05.874759       1 manifestwork_controller.go:179] Reconcile work test-1-klusterlet-addon-workmgr fails with err: Failed to update work status with err Get "https://api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:05.874887       1 base_controller.go:231] "ManifestWorkAgent" controller failed to sync "test-1-klusterlet-addon-workmgr", err: Failed to update work status with err Get "api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks/test-1-klusterlet-addon-workmgr": x509: certificate signed by unknown authority
E0917 03:04:37.245859       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156: Failed to watch *v1.ManifestWork: failed to list *v1.ManifestWork: Get "api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/namespaces/test-1/manifestworks?resourceVersion=607424": x509: certificate signed by unknown authority
```

以下の **registration-agent** ログを確認してください。

```
I0917 02:27:41.525026       1 event.go:282] Event(v1.ObjectReference{Kind:"Namespace", Namespace:"open-cluster-management-agent", Name:"open-cluster-management-agent", UID:"", APIVersion:"v1", ResourceVersion:"", FieldPath:""}): type: 'Normal' reason: 'ManagedClusterAvailableConditionUpdated' update managed cluster "test-1" available condition to "True", due to "Managed cluster is available"
E0917 02:58:26.315984       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156: Failed to watch *v1beta1.CertificateSigningRequest: Get "https://api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true"": x509: certificate signed by unknown authority
E0917 02:58:26.598343       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156: Failed to watch *v1.ManagedCluster: Get "https://api.aaa-ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
```

```
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true": x509: certificate
signed by unknown authority
E0917 02:58:27.613963       1 reflector.go:127] k8s.io/client-go@v0.19.0/tools/cache/reflector.go:156:
Failed to watch *v1.ManagedCluster: failed to list *v1.ManagedCluster: Get "https://api.aaa-
ocp.dev02.location.com:6443/apis/cluster.management.io/v1/managedclusters?
allowWatchBookmarks=true&fieldSelector=metadata.name%3Dtest-
1&resourceVersion=607408&timeout=9m33s&timeoutSeconds=573&watch=true"": x509: certificate
signed by unknown authority
```

1.9.11.3. 問題の解決: 証明書の変更後にクラスターがオフラインになる

マネージドクラスターが **local-cluster** の場合、または multicluster engine operator でマネージドクラスターが作成された場合に、マネージドクラスターを再インポートするには 10 分以上待つ必要があります。

マネージドクラスターをすぐに再インポートするには、ハブクラスター上のマネージドクラスターのインポートシークレットを削除し、multicluster engine operator を使用して復元します。以下のコマンドを実行します。

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

<cluster_name> は、復元するマネージドクラスターの名前に置き換えます。

multicluster engine operator を使用してインポートされたマネージドクラスターを再インポートする場合は、次の手順を実行して、マネージドクラスターを復元します。

1. ハブクラスターで、次のコマンドを実行してマネージドクラスターのインポートシークレットを再作成します。

   ```
   oc delete secret -n <cluster_name> <cluster_name>-import
   ```

   <cluster_name> を、インポートするマネージドクラスターの名前に置き換えます。

2. ハブクラスターで、次のコマンドを実行して、マネージドクラスターのインポートシークレットを YAML ファイルに公開します。

   ```
   oc get secret -n <cluster_name> <cluster_name>-import -ojsonpath='{.data.import\.yaml}' |
   base64 --decode  > import.yaml
   ```

   <cluster_name> を、インポートするマネージドクラスターの名前に置き換えます。

3. マネージドクラスターで、次のコマンドを実行して **import.yaml** ファイルを適用します。

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

注記: 前の手順では、マネージドクラスターがハブクラスターから切り離されません。この手順により、必要なマニフェストがマネージドクラスターの現在の設定 (新しい証明書情報を含む) で更新されます。

1.9.12. クラスターのステータスが offline から available に変わる場合のトラブルシューティング

マネージドクラスターのステータスは、環境またはクラスターを手動で変更することなく、**offline** と **available** との間で切り替わります。

1.9.12.1. 現象: クラスターのステータスが offline から available に変わる

マネージドクラスターからハブクラスターへのネットワーク接続が不安定な場合に、マネージドクラスターのステータスが **offline** と **available** との間で順に切り替わると、ハブクラスターにより報告されます。

1.9.12.2. 問題の解決: クラスターのステータスが offline から available に変わる

この問題を解決するには、以下の手順を実行します。

1. 次のコマンドを入力して、ハブクラスターで **ManagedCluster** の仕様を編集します。

   ```
   oc edit managedcluster <cluster-name>
   ```

   cluster-name は、マネージドクラスターの名前に置き換えます。

2. **ManagedCluster** 仕様の **leaseDurationSeconds** の値を増やします。デフォルト値は 5 分ですが、ネットワークの問題がある状態で接続を維持するには十分でない場合があります。リースの時間を長く指定します。たとえば、設定を 20 分に増やします。

1.9.13. VMware vSphere でのクラスター作成のトラブルシューティング

VMware vSphere で Red Hat OpenShift Container Platform クラスターを作成する時に問題が発生した場合は、以下のトラブルシューティング情報を参照して、この情報のいずれかが問題に対応しているかどうかを確認します。

注記: VMware vSphere でクラスター作成プロセスが失敗した場合に、リンクが有効にならずログが表示されないことがあります。上記が発生する場合は、**hive-controllers** Pod のログを確認して問題を特定できます。**hive-controllers** ログは **hive** namespace にあります。

1.9.13.1. 証明書 の IP SAN エラーでマネージドクラスターの作成に失敗する

1.9.13.1.1. 現象: 証明書の IP SAN エラーでマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、証明書 IP SAN エラーを示すエラーメッセージでクラスターに問題が発生します。

1.9.13.1.2. 問題の特定: 証明書の IP SAN エラーでマネージドクラスターの作成に失敗する

マネージドクラスターのデプロイメントに失敗して、デプロイメントログに以下のエラーが返されます。

```
time="2020-08-07T15:27:55Z" level=error msg="Error: error setting up new vSphere SOAP client: Post https://147.1.1.1/sdk: x509: cannot validate certificate for xx.xx.xx.xx because it doesn't contain any IP SANs"
time="2020-08-07T15:27:55Z" level=error
```

1.9.13.1.3. 問題の解決: 証明書の IP SAN エラーでマネージドクラスターの作成に失敗する

認証情報の IP アドレスではなく VMware vCenter サーバー完全修飾ホスト名を使用します。また、VMware vCenter CA 証明書を更新して、IP SAN を組み込むこともできます。

1.9.13.2. 不明な証明局のエラーでマネージドクラスターの作成に失敗する

1.9.13.2.1. 現象: 不明な証明局のエラーでマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、証明書が不明な証明局により署名されているのでクラスターに問題が発生します。

1.9.13.2.2. 問題の特定: 不明な証明局のエラーでマネージドクラスターの作成に失敗する

マネージドクラスターのデプロイメントに失敗して、デプロイメントログに以下のエラーが返されます。

```
Error: error setting up new vSphere SOAP client: Post https://vspherehost.com/sdk: x509: certificate signed by unknown authority"
```

1.9.13.2.3. 問題の解決: 不明な証明局のエラーでマネージドクラスターの作成に失敗する

認証情報の作成時に認証局の正しい証明書が入力されていることを確認します。

1.9.13.3. 証明書の期限切れでマネージドクラスターの作成に失敗する

1.9.13.3.1. 現象: 証明書の期限切れでマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、証明書の期限が切れているか、有効にしていないため、クラスターに問題が発生します。

1.9.13.3.2. 問題の特定: 証明書の期限切れでマネージドクラスターの作成に失敗する

マネージドクラスターのデプロイメントに失敗して、デプロイメントログに以下のエラーが返されます。

```
x509: certificate has expired or is not yet valid
```

1.9.13.3.3. 問題の解決: 証明書の期限切れでマネージドクラスターの作成に失敗する

ESXi ホストの時間が同期されていることを確認します。

1.9.13.4. タグ付けの権限が十分ではないためマネージドクラスターの作成に失敗する

1.9.13.4.1. 現象: タグ付けの権限が十分ではないためマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、タグ付けの使用に十分な権限がないためクラスターに問題が発生します。

1.9.13.4.2. 問題の特定: タグ付けの権限が十分にないためにマネージドクラスターの作成に失敗する

マネージドクラスターのデプロイメントに失敗して、デプロイメントログに以下のエラーが返されます。

```
time="2020-08-07T19:41:58Z" level=debug msg="vsphere_tag_category.category: Creating..."
time="2020-08-07T19:41:58Z" level=error
time="2020-08-07T19:41:58Z" level=error msg="Error: could not create category: POST
https://vspherehost.com/rest/com/vmware/cis/tagging/category: 403 Forbidden"
time="2020-08-07T19:41:58Z" level=error
```

```
time="2020-08-07T19:41:58Z" level=error msg="  on ../tmp/openshift-install-436877649/main.tf line
54, in resource \"vsphere_tag_category\" \"category\":"
time="2020-08-07T19:41:58Z" level=error msg="  54: resource \"vsphere_tag_category\" \"category\"
{"
```

1.9.13.4.3. 問題の解決: タグ付けの権限が十分ではないためマネージドクラスターの作成に失敗する

VMware vCenter が必要とするアカウントの権限が正しいことを確認します。詳細は、インストール時に削除されたイメージレジストリー を参照してください。

1.9.13.5. 無効な dnsVIP でマネージドクラスターの作成に失敗する

1.9.13.5.1. 現象: 無効な dnsVIP でマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、dnsVIP が無効であるため、クラスターに問題が発生します。

1.9.13.5.2. 問題の特定: 無効な dnsVIP でマネージドクラスターの作成に失敗する

VMware vSphere で新しいマネージドクラスターをデプロイしようとして以下のメッセージが表示されるのは、VMware Installer Provisioned Infrastructure (IPI) をサポートしない以前の OpenShift Container Platform リリースイメージを使用しているためです。

```
failed to fetch Master Machines: failed to load asset \\\"Install Config\\\": invalid \\\"install-
config.yaml\\\" file: platform.vsphere.dnsVIP: Invalid value: \\\"\\\": \\\"\\\" is not a valid IP
```

1.9.13.5.3. 問題の解決: 無効な dnsVIP でマネージドクラスターの作成に失敗する

VMware インストーラーでプロビジョニングされるインフラストラクチャーをサポートする OpenShift Container Platform で、新しいバージョンのリリースイメージを選択します。

1.9.13.6. ネットワークタイプが正しくないためマネージドクラスターの作成に失敗する

1.9.13.6.1. 現象: ネットワークタイプが正しくないためマネージドクラスターの作成に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、間違ったネットワークタイプが指定されているため、クラスターに問題が発生します。

1.9.13.6.2. 問題の特定: ネットワークタイプが正しくないためマネージドクラスターの作成に失敗する

VMware vSphere で新しいマネージドクラスターをデプロイしようとして以下のメッセージが表示されるのは、VMware Installer Provisioned Infrastructure (IPI) をサポートしない以前の OpenShift Container Platform イメージを使用しているためです。

```
time="2020-08-11T14:31:38-04:00" level=debug msg="vsphereprivate_import_ova.import:
Creating..."
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error msg="Error: rpc error: code = Unavailable desc =
transport is closing"
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=error
time="2020-08-11T14:31:39-04:00" level=fatal msg="failed to fetch Cluster: failed to generate asset
\"Cluster\": failed to create cluster: failed to apply Terraform: failed to complete the change"
```

-

1.9.13.6.3. 問題の解決: ネットワークタイプが正しくないためマネージドクラスターの作成に失敗する

指定の VMware クラスターに対して有効な VMware vSphere ネットワークタイプを選択します。

1.9.13.7. ディスクの変更処理のエラーでマネージドクラスターの作成に失敗する

1.9.13.7.1. 現象: ディスクの変更処理のエラーが原因でマネージドクラスターの追加に失敗する

VMware vSphere で新規の Red Hat OpenShift Container Platform クラスターを作成した後に、ディスク変更処理時にエラーによりクラスターに問題が発生します。

1.9.13.7.2. 問題の特定: ディスクの変更処理のエラーが原因でマネージドクラスターの追加に失敗する

以下のようなメッセージがログに表示されます。

```
ERROR
ERROR Error: error reconfiguring virtual machine: error processing disk changes post-clone: disk.0:
ServerFaultCode: NoPermission: RESOURCE (vm-71:2000), ACTION (queryAssociatedProfile):
RESOURCE (vm-71), ACTION (PolicyIDByVirtualDisk)
```

1.9.13.7.3. 問題の解決: ディスクの変更処理のエラーが原因でマネージドクラスターの追加に失敗する

VMware vSphere クライアントを使用してユーザーに **プロファイル駆動型のストレージ権限** の **全権限** を割り当てます。

1.9.14. ステータスが Pending または Failed のクラスターのコンソールでのトラブルシューティング

作成してたクラスターのステータスがコンソールで **Pending** または **Failed** と表示されている場合は、以下の手順を実行して問題のトラブルシューティングを実行します。

1.9.14.1. 現象: コンソールでステータスが Pending または Failed のクラスターのトラブルシューティング

コンソールを使用して新しいクラスターを作成した後、クラスターは **Pending** のステータスを超えて進行しないか、**Failed** ステータスを表示します。

1.9.14.2. 問題の特定: コンソールでステータスが Pending または Failed のクラスター

クラスターのステータスが **Failed** と表示される場合は、クラスターの詳細ページに移動して、提供されたログへのリンクに進みます。ログが見つからない場合や、クラスターのステータスが **Pending** と表示される場合は、以下の手順を実行してログを確認します。

- 手順 1

    1. ハブクラスターで以下のコマンドを実行し、新規クラスターの namespace に作成した Kubernetes Pod の名前を表示します。

        ```
        oc get pod -n <new_cluster_name>
        ```

        new_cluster_name は、作成したクラスター名に置き換えます。

2. 名前に **provision** の文字列が含まれる Pod が表示されていない場合は、手順 2 に進みます。タイトルに **provision** が含まれる Pod があった場合は、ハブクラスターで以下のコマンドを実行して、その Pod のログを表示します。

```
oc logs <new_cluster_name_provision_pod_name> -n <new_cluster_name> -c hive
```

new_cluster_name_provision_pod_name は、作成したクラスター名の後に **provision** が含まれる Pod 名を指定するように置き換えます。

3. ログでエラーを検索してください。この問題の原因が解明する場合があります。

- 手順 2
名前に **provision** が含まれる Pod がない場合は、問題がプロセスの初期段階で発生しています。ログを表示するには、以下の手順を実行します。

1. ハブクラスターで以下のコマンドを実行してください。

```
oc describe clusterdeployments -n <new_cluster_name>
```

new_cluster_name は、作成したクラスター名に置き換えます。クラスターのインストールログの詳細は、Red Hat OpenShift ドキュメントの インストールログの収集 を参照してください。

2. リソースの **Status.Conditions.Message** と **Status.Conditions.Reason** のエントリーに問題に関する追加の情報があるかどうかを確認します。

1.9.14.3. 問題の解決: コンソールでステータスが Pending または Failed のクラスター

ログでエラーを特定した後に、エラーの解決方法を決定してから、クラスターを破棄して、作り直してください。

以下の例では、サポート対象外のゾーンを選択している可能性を示すログエラーと、解決に必要なアクションが提示されています。

```
No subnets provided for zones
```

クラスターの作成時に、サポートされていないリージョンにあるゾーンを 1 つ以上選択しています。問題解決用にクラスターを再作成する時に、以下のアクションの 1 つを実行します。

- リージョン内の異なるゾーンを選択します。
- 他のゾーンをリストしている場合は、サポートを提供しないゾーンを省略します。
- お使いのクラスターに、別のリージョンを選択します。

ログから問題を特定した後に、クラスターを破棄し、再作成します。

クラスター作成の詳細は、クラスター作成の概要 を参照してください。

1.9.15. OpenShift Container Platform バージョン 3.11 クラスターのインポートの失敗時のトラブルシューティング

1.9.15.1. 現象: OpenShift Container Platform バージョン 3.11 クラスターのインポートに失敗する

Red Hat OpenShift Container Platform バージョン 3.11 クラスターのインポートを試行すると、以下の内容のようなログメッセージでインポートに失敗します。

```
customresourcedefinition.apiextensions.k8s.io/klusterlets.operator.open-cluster-management.io
configured
clusterrole.rbac.authorization.k8s.io/klusterlet configured
clusterrole.rbac.authorization.k8s.io/open-cluster-management:klusterlet-admin-aggregate-clusterrole
configured
clusterrolebinding.rbac.authorization.k8s.io/klusterlet configured
namespace/open-cluster-management-agent configured
secret/open-cluster-management-image-pull-credentials unchanged
serviceaccount/klusterlet configured
deployment.apps/klusterlet unchanged
klusterlet.operator.open-cluster-management.io/klusterlet configured
Error from server (BadRequest): error when creating "STDIN": Secret in version "v1" cannot be
handled as a Secret:
v1.Secret.ObjectMeta:
v1.ObjectMeta.TypeMeta: Kind: Data: decode base64: illegal base64 data at input byte 1313, error
found in #10 byte of ...|dhruy45="},"kind":"|..., bigger context
...|tye56u56u568yuo7i67i67i67o556574i"},"kind":"Secret","metadata":{"annotations":{"kube|...
```

1.9.15.2. 問題の特定: OpenShift Container Platform バージョン 3.11 クラスターのインポートに失敗する

この問題は多くの場合、インストールされている **kubectl** コマンドラインツールのバージョンが 1.11 以前であるために発生します。以下のコマンドを実行して、実行中の **kubectl** コマンドラインツールのバージョンを表示します。

```
kubectl version
```

返されたデータがバージョンが 1.11 以前の場合は、**問題の解決: OpenShift Container Platform バージョン 3.11 クラスターのインポートに失敗する** に記載される修正のいずれかを実行します。

1.9.15.3. 問題の解決: OpenShift Container Platform バージョン 3.11 クラスターのインポートに失敗する

この問題は、以下のいずれかの手順を実行して解決できます。

- 最新バージョンの **kubectl** コマンドラインツールをインストールします。
  1. **kubectl** ツールの最新バージョンを、Kubernetes ドキュメントの kubectl のインストールとセットアップ からダウンロードします。
  2. **kubectl** ツールのアップグレード後にクラスターを再度インポートします。
- import コマンドが含まれるファイルを実行します。
  1. CLI を使用したマネージドクラスターのインポート の手順を開始します。
  2. クラスターの import コマンドを作成する場合には、この import コマンドを **import.yaml** という名前の YAML ファイルにコピーします。
  3. 以下のコマンドを実行して、ファイルからクラスターを再度インポートします。

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

1.9.16. degraded 状態にある Klusterlet のトラブルシューティング

Klusterlet の状態が Degraded の場合は、マネージドクラスターの Klusterlet エージェントの状態を診断しやすくなります。Klusterlet の状態が Degraded になると、マネージドクラスターの Klusterlet エージェントで発生する可能性のあるエラーに対応する必要があります。Klusterlet の degraded の状態が **True** に設定されている場合は、以下の情報を参照します。

1.9.16.1. 現象: Klusterlet の状態が degraded である

マネージドクラスターで Klusterlet をデプロイした後に、**KlusterletRegistrationDegraded** または **KlusterletWorkDegraded** の状態が True と表示されます。

1.9.16.2. 問題の特定: Klusterlet の状態が degraded である

1. マネージドクラスターで以下のコマンドを実行して、Klusterlet のステータスを表示します

```
kubectl get klusterlets klusterlet -oyaml
```

2. **KlusterletRegistrationDegraded** または **KlusterletWorkDegraded** をチェックして、状態が **True** に設定されいるかどうかを確認します。記載されている Degraded の状態は、**問題の解決** に進みます。

1.9.16.3. 問題の解決: Klusterlet の状態が degraded である

ステータスが Degraded のリストおよびこれらの問題の解決方法を参照してください。

- **KlusterletRegistrationDegraded** の状態が True で、この状態の理由が BootStrapSecretMissing の場合は、**open-cluster-management-agent** namespace にブートストラップのシークレットを作成する必要があります。
- **KlusterletRegistrationDegraded** の状態が True と表示され、状態の理由が BootstrapSecretError または BootstrapSecretUnauthorized の場合は、現在のブートストラップシークレットが無効です。現在のブートストラップシークレットを削除して、**open-cluster-management-agent** namespace で有効なブートストラップシークレットをもう一度作成します。
- **KlusterletRegistrationDegraded** および **KlusterletWorkDegraded** が True と表示され、状態の理由が HubKubeConfigSecretMissing の場合は、Klusterlet を削除して作成し直します。
- **KlusterletRegistrationDegraded** および **KlusterletWorkDegraded** が True と表示され、状態の理由が ClusterNameMissing、KubeConfigMissing、HubConfigSecretError、または HubConfigSecretUnauthorized の場合は、**open-cluster-management-agent** namespace からハブクラスターの kubeconfig シークレットを削除します。登録エージェントは再度ブートストラップして、新しいハブクラスターの kubeconfig シークレットを取得します。
- **KlusterletRegistrationDegraded** が True と表示され、状態の理由が GetRegistrationDeploymentFailed または UnavailableRegistrationPod の場合は、状態のメッセージを確認して、問題の詳細を取得して解決してみてください。
- **KlusterletWorkDegraded** が True と表示され、状態の理由が GetWorkDeploymentFailed または UnavailableWorkPod の場合は、状態のメッセージを確認して、問題の詳細を取得し、解決してみてください。

1.9.17. クラスターの削除後も namespace が残る

マネージドクラスターを削除すると、通常 namespace はクラスターの削除プロセスの一部として削除されます。まれに namespace は一部のアーティファクトが含まれた状態で残る場合があります。このような場合は、namaspace を手動で削除する必要があります。

1.9.17.1. 現象: クラスターの削除後も namespace が残る

マネージドクラスターの削除後に namespace が削除されません。

1.9.17.2. 問題の解決: クラスターの削除後も namespace が残る

namespace を手作業で削除するには、以下の手順を実行します。

1. 次のコマンドを実行して、<cluster_name> namespace に残っているリソースのリストを作成します。

   ```
   oc api-resources --verbs=list --namespaced -o name | grep -E
   '^secrets|^serviceaccounts|^managedclusteraddons|^roles|^rolebindings|^manifestworks|^lease
   |^managedclusterinfo|^appliedmanifestworks'|^clusteroauths' | xargs -n 1 oc get --show-kind -
   -ignore-not-found -n <cluster_name>
   ```

   cluster_name は、削除を試みたクラスターの namespace 名に置き換えます。

2. 以下のコマンドを入力してリストを編集し、ステータスが **Delete** ではないリストから特定したリソースを削除します。

   ```
   oc edit <resource_kind> <resource_name> -n <namespace>
   ```

   resource_kind は、リソースの種類に置き換えます。**resource_name** は、リソース名に置き換えます。**namespace** は、リソースの namespace に置き換えます。

3. メタデータで **finalizer** 属性の場所を特定します。

4. vi エディターの **dd** コマンドを使用して、Kubernetes 以外のファイナライザーを削除します。

5. **:wq** コマンドを入力し、リストを保存して **vi** エディターを終了します。

6. 以下のコマンドを入力して namespace を削除します。

   ```
   oc delete ns <cluster-name>
   ```

   cluster-name を、削除する namespace の名前に置き換えます。

1.9.18. クラスターのインポート時の auto-import-secret-exists エラー

クラスターのインポートは、auto import secret exists というエラーメッセージで失敗します。

1.9.18.1. 現象: クラスターのインポート時の Auto-import-secret-exists エラー

管理用のハイブクラスターをインポートすると、**auto-import-secret already exists** というエラーが表示されます。

1.9.18.2. 問題の解決: クラスターのインポート時の Auto-import-secret-exists エラー

この問題は、以前に管理されていたクラスターをインポートしようとすると発生します。これが生じると、クラスターを再インポートしようとすると、シークレットは競合します。

この問題を回避するには、以下の手順を実行します。

1. 既存の **auto-import-secret** を手動で削除するには、ハブクラスターで以下のコマンドを実行します。

   ```
   oc delete secret auto-import-secret -n <cluster-namespace>
   ```

   namespace は、お使いのクラスターの namespace に置き換えます。

2. クラスターインポートの概要 の手順を使用して、クラスターを再度インポートします。

1.9.19. Troubleshooting missing PlacementDecision after creating Placement

Placement の作成後に **PlacementDescision** が生成されない場合は、手順に従って問題をトラブルシューティングしてください。

1.9.19.1. 事象: Placement の作成後に PlacementDecision が見つからない

Placement を作成した後、**PlacementDescision** は自動的に生成されません。

1.9.19.2. 問題の解決: Placement の作成後に PlacementDecision が見つからない

この問題を解決するには、以下の手順を実行します。

1. 次のコマンドを実行して **Placement** 条件を確認します。

   ```
   kubectl describe placement <placement-name>
   ```

   placement-name を **Placement** の名前に置き換えます。

   出力は次の例のような内容になります。

   ```
   Name:         demo-placement
   Namespace:    default
   Labels:       <none>
   Annotations:  <none>
   API Version:  cluster.open-cluster-management.io/v1beta1
   Kind:         Placement
   Status:
     Conditions:
       Last Transition Time:       2022-09-30T07:39:45Z
       Message:                    Placement configurations check pass
       Reason:                     Succeedconfigured
       Status:                     False
       Type:                       PlacementMisconfigured
       Last Transition Time:       2022-09-30T07:39:45Z
       Message:                    No valid ManagedClusterSetBindings found in placement
   namespace
       Reason:                     NoManagedClusterSetBindings
       Status:                     False
       Type:                       PlacementSatisfied
     Number Of Selected Clusters:  0
   ```

2. **PlacementMisconfigured** および **PlacementSatisfied** の **Status** の出力を確認します。

   - **PlacementMisconfigured Status** が true の場合、**Placement** に設定エラーがあります。設定エラーの詳細とその解決方法については、含まれているメッセージを確認してください。

   - **PlacementSatisfied Status** が false の場合、**Placement** を満たすマネージドクラスターはありません。詳細とエラーの解決方法については、含まれているメッセージを確認してください。前の例では、placement namespace に **ManagedClusterSetBindings** が見つかりませんでした。

3. **Events** で各クラスターのスコアを確認して、スコアの低い一部のクラスターが選択されていない理由を確認できます。出力は次の例のような内容になります。

```
Name:         demo-placement
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  cluster.open-cluster-management.io/v1beta1
Kind:         Placement
Events:
  Type    Reason          Age   From                Message
  ----    ------          ----  ----                -------
  Normal  DecisionCreate  2m10s   placementController  Decision demo-placement-decision-1 is created with placement demo-placement in namespace default
  Normal  DecisionUpdate  2m10s   placementController  Decision demo-placement-decision-1 is updated with placement demo-placement in namespace default
  Normal  ScoreUpdate     2m10s   placementController  cluster1:0 cluster2:100 cluster3:200
  Normal  DecisionUpdate  3s      placementController  Decision demo-placement-decision-1 is updated with placement demo-placement in namespace default
  Normal  ScoreUpdate     3s      placementController  cluster1:200 cluster2:145 cluster3:189 cluster4:200
```

注記: 配置コントローラーはスコアを割り当て、フィルター処理された **ManagedCluster** ごとにイベントを生成します。クラスタースコアが変化すると、配置コントローラーは新しいイベントを生成します。

1.9.20. Dell ハードウェアにおけるベアメタルホストの検出エラーのトラブルシューティング

Dell ハードウェアでベアメタルホストの検出が失敗した場合、Integrated Dell Remote Access Controller (iDRAC) が不明な認証局からの証明書を許可しないように設定されている可能性があります。

1.9.20.1. 現象: Dell ハードウェアでのベアメタルホストの検出エラー

ベースボード管理コントローラーを使用してベアメタルホストを検出する手順を完了すると、次のようなエラーメッセージが表示されます。

```
ProvisioningError 51s metal3-baremetal-controller Image provisioning failed: Deploy step deploy.deploy failed with BadRequestError: HTTP POST https://<bmc_address>/redfish/v1/Managers/iDRAC.Embedded.1/VirtualMedia/CD/Actions/VirtualMedia.InsertMedia returned code 400. Base.1.8.GeneralError: A general error has occurred. See ExtendedInfo for more information Extended information: [
{"Message": "Unable to mount remote share https://<ironic_address>/redfish/boot-<uuid>.iso.",
```

'MessageArgs': ["https://<ironic_address>/redfish/boot-<uuid>.iso"], "MessageArgs@odata.count": 1, "MessageId": "IDRAC.2.5.RAC0720", "RelatedProperties": ["#/Image"], "RelatedProperties@odata.count": 1, "Resolution": "Retry the operation.", "Severity": "Informational"}]

1.9.20.2. 問題の解決: Dell ハードウェアでのベアメタルホストの検出の失敗

iDRAC は、不明な認証局からの証明書を受け入れないように設定されています。

この問題を回避するには、次の手順を実行して、ホスト iDRAC のベースボード管理コントローラーで証明書の検証を無効にします。

1. iDRAC コンソールで、 **Configuration** > **Virtual media** > **Remote file share** に移動します。
2. **Expired or invalid certificate action** の値を **Yes** に変更します。

1.9.21. 最小限の ISO 起動エラーのトラブルシューティング

最小限の ISO を起動しようとすると問題が発生する可能性があります。

1.9.21.1. 症状: 最小限の ISO ブート失敗

ブート画面には、ホストがルートファイルシステムイメージのダウンロードに失敗したことが示されます。

1.9.21.2. 問題の解決: ISO ブートの失敗が最小限に抑えられる

問題のトラブルシューティング方法は、OpenShift Container Platform の Assisted Installer ドキュメントの 最小限の ISO ブート失敗のトラブルシューティング を参照してください。

1.9.22. Red Hat OpenShift Virtualization 上のホステッドクラスターのトラブルシューティング

Red Hat OpenShift Virtualization でホストされたクラスターのトラブルシューティングを行う場合は、トップレベルの **HostedCluster** リソースと **NodePool** リソースから始めて、根本原因が見つかるまでスタックを下位に向かって作業します。以下の手順は、一般的な問題の根本原因を検出するのに役立ちます。

1.9.22.1. 現象: HostedCluster リソースが部分的な状態でスタックする

HostedCluster リソースが保留中であるため、Hosted control plane が完全にオンラインになりません。

1.9.22.1.1. 問題の特定: 前提条件、リソースの状態、ノードと Operator のステータスを確認します。

- Red Hat OpenShift Virtualization 上のホステッドクラスターのすべての前提条件を満たしていることを確認します。
- 進行を妨げる検証エラーがないか、**HostedCluster** リソースと **NodePool** リソースの状態を表示します。
- ホステッドクラスターの **kubeconfig** ファイルを使用して、ホステッドクラスターのステータスを検査します。

- **oc get clusteroperators** コマンドの出力を表示して、保留中のクラスター Operator を表示します。
- **oc get nodes** コマンドの出力を表示して、ワーカーノードの準備ができていることを確認します。

1.9.22.2. 現象: ワーカーノードが登録されない

Hosted control plane にはワーカーノードが登録されていないため、 Hosted control plane は完全にオンラインになりません。

1.9.22.2.1. 問題の特定: Hosted control plane のさまざまな部分のステータスを確認します。

- **HostedCluster** と **NodePool** の障害の状態を表示して、問題の内容を示します。
- 次のコマンドを入力して、**NodePool** リソースの KubeVirt ワーカーノード仮想マシン (VM) のステータスを表示します。

```
oc get vm -n <namespace>
```

- 仮想マシンがプロビジョニング状態でスタックしている場合は、次のコマンドを入力して、VM namespace 内の CDI インポート Pod を表示し、インポーター Pod が完了していない理由を調べます。

```
oc get pods -n <namespace> | grep "import"
```

- VM が開始状態でスタックしている場合は、次のコマンドを入力して、virt-launcher Pod のステータスを表示します。

```
oc get pods -n <namespace> -l kubevirt.io=virt-launcher
```

  virt-launcher Pod が保留状態にある場合は、Pod がスケジュールされていない理由を調査してください。たとえば、virt-launcher Pod の実行に必要なリソースが十分存在しない可能性があります。

- 仮想マシンが実行されていてもワーカーノードとして登録されていない場合は、Web コンソールを使用して、影響を受ける VM の 1 つへの VNC アクセスを取得します。VNC 出力は ignition 設定が適用されたかどうかを示します。仮想マシンが起動時に Hosted control plane Ignition サーバーにアクセスできない場合、仮想マシンは正しくプロビジョニングできません。
- Ignition 設定が適用されても、仮想マシンがノードとして登録されていない場合は、問題の特定: 仮想マシンコンソールログへのアクセス で、起動時に仮想マシンコンソールログにアクセスする方法を確認してください。

1.9.22.3. 現象: ワーカーノードが NotReady 状態でスタックする

クラスターの作成中、ネットワークスタックがロールアウトされる間、ノードは一時的に **NotReady** 状態になります。プロセスのこの部分は正常です。ただし、プロセスのこの部分に 15 分以上かかる場合は、問題が発生している可能性があります。

1.9.22.3.1. 問題の特定: ノードオブジェクトと Pod を調査する

- 次のコマンドを入力して、ノードオブジェクトの状態を表示し、ノードの準備ができていない理由を特定します。

```
oc get nodes -o yaml
```

- 次のコマンドを入力して、クラスター内で障害が発生している Pod を探します。

```
oc get pods -A --field-selector=status.phase!=Running,status,phase!=Succeeded
```

1.9.22.4. 現象: Ingress およびコンソールクラスター Operator がオンラインにならない

Ingress およびコンソールクラスター Operator がオンラインではないため、Hosted control plane は完全にはオンラインになりません。

1.9.22.4.1. 問題の特定: ワイルドカード DNS ルートとロードバランサーを確認する

- クラスターがデフォルトの Ingress 動作を使用する場合は、次のコマンドを入力して、仮想マシン (VM) がホストされている OpenShift Container Platform クラスターでワイルドカード DNS ルートが有効になっていることを確認します。

```
oc patch ingresscontroller -n openshift-ingress-operator default --type=json -p '[{ "op": "add", "path": "/spec/routeAdmission", "value": {wildcardPolicy: "WildcardsAllowed"}}]'
```

- Hosted control plane にカスタムベースドメインを使用する場合は、次の手順を実行します。
  - ロードバランサーが VM Pod を正しくターゲットにしていることを確認してください。
  - ワイルドカード DNS エントリーがロードバランサー IP をターゲットにしていることを確認してください。

1.9.22.5. 現象: ホステッドクラスターのロードバランサーサービスが利用できない

ロードバランサーサービスは利用できないため、Hosted control plane は完全にオンラインになりません。

1.9.22.5.1. 問題の特定: イベント、詳細、kccm Pod を確認します。

- ホステッドクラスター内のロードバランサーサービスに関連付けられたイベントおよび詳細を探します。
- デフォルトでは、ホステッドクラスターのロードバランサーは、Hosted control plane の namespace 内の kubevirt-cloud-controller-manager によって処理されます。kccm Pod がオンラインであることを確認し、そのログにエラーや警告がないか確認してください。Hosted control plane namespace で kccm Pod を特定するには、次のコマンドを入力します。

```
oc get pods -n <hosted-control-plane-namespace> -l app=cloud-controller-manager
```

1.9.22.6. 現象: ホステッドクラスター PVC が使用できない

ホステッドクラスターの永続ボリューム要求 (PVC) が使用できないため、Hosted control plane は完全にはオンラインになりません。

1.9.22.6.1. 問題の特定：PVC イベントおよび詳細、およびコンポーネントログを確認します。

- PVC に関連するイベントと詳細を探して、どのエラーが発生しているかを把握します。

- PVC が Pod に割り当てられていない場合、ホステッドクラスター内の kubevirt-csi-node daemonset コンポーネントのログを表示して問題をさらに調査します。各ノードの kubevirt-csi-node Pod を識別するには、次のコマンドを入力します。

```
oc get pods -n openshift-cluster-csi-drivers -o wide -l app=kubevirt-csi-driver
```

- PVC が永続ボリューム (PV) にバインドできない場合は、Hosted control plane の namespace 内の kubevirt-csi-controller コンポーネントのログを表示します。Hosted control plane の namespace 内の kubevirt-csi-controller Pod を識別するには、次のコマンドを入力します。

```
oc get pods -n <hcp namespace> -l app=kubevirt-csi-driver
```

1.9.22.7. 現象: 仮想マシンノードがクラスターに正しく参加していない。

仮想マシンノードがクラスターに正しく参加していないため、Hosted control plane が完全にオンラインになりません。

1.9.22.7.1. 問題の特定: 仮想マシンコンソールログにアクセスします。

仮想マシンコンソールログにアクセスするには、How to get serial console logs for VMs part of OpenShift Virtualization Hosted Control Plane clusters の手順を実行します。