Red Hat Advanced Cluster Management for Kubernetes 2.6

multicluster engine

Read more to learn how to create and manage the multicluster engine for Kubernetes.

Last Updated: 2023-10-31

Red Hat Advanced Cluster Management for Kubernetes 2.6 multicluster engine

Read more to learn how to create and manage the multicluster engine for Kubernetes.

Legal Notice

Abstract

Read more to learn how to create and manage the multicluster engine for Kubernetes.

Table of Contents

CHAPTER 1. THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR OVERVIEW 13
1.1. ABOUT THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR 16
1.1.1. Requirements and recommendations 16
1.1.1.1. Supported browsers and platforms 16
1.1.2. Networking 17
1.1.2.1. Hub cluster network configuration 17
1.1.2.2. Managed cluster network configuration 17
1.1.2.3. Additional networking requirements when installing using the infrastructure operator 18
1.1.2.4. Additional networking requirements when installing using the Hive Operator 18
1.1.2.4.1. Hosted control planes networking requirements (Technology Preview) 19
1.1.3. Console overview 19
1.1.4. Role-based access control 20
1.1.4.1. Overview of roles 20
1.1.4.1.1. Table of role definition 20
1.1.4.1.2. Cluster lifecycle RBAC 22
1.1.4.1.2.1. Cluster pools RBAC 23
1.1.4.1.2.2. Console RBAC table for cluster lifecycle 24
1.1.4.1.2.3. Table of RBAC API table for cluster lifecycle 24
1.1.4.1.2.4. Credentials role-based access control 26
1.2. INSTALL 26
1.2.1. Installing while connected online 27
1.2.1.1. Prerequisites 28
1.2.1.2. Confirm your OpenShift Container Platform installation 28
1.2.1.3. Installing from the OperatorHub web console interface 29
1.2.1.4. Installing from the OpenShift Container Platform CLI 30
1.2.1.5. Installing on infrastructure nodes 32
1.2.1.5.1. Add infrastructure nodes to the OpenShift Container Platform cluster 32
1.2.1.5.2. Operator Lifecycle Manager Subscription additional configuration 32
1.2.1.5.3. MultiClusterEngine custom resource additional configuration 32
1.2.2. Install on disconnected networks 33
1.2.2.1. Prerequisites 33
1.2.2.2. Confirm your OpenShift Container Platform installation 33
1.2.2.3. Installing in a disconnected environment 34
1.2.3. Advanced configuration 36
1.2.3.1. Local-cluster enablement 36
1.2.3.2. Custom image pull secret 36
1.2.3.3. Target namespace 37
1.2.3.4. availabilityConfig 37
1.2.3.5. nodeSelector 38
1.2.3.6. tolerations 38
1.2.3.7. ManagedServiceAccount add-on (Technology Preview) 38
1.2.3.8. Hypershift add-on (Technology Preview) 39
1.2.4. Uninstalling 39
1.2.4.1. Prerequisite: Detach enabled services 39
1.2.4.2. Removing resources by using commands 40
1.2.4.3. Deleting the components by using the console 40
1.2.4.4. Troubleshooting Uninstall 41
1.3. MANAGING CREDENTIALS 41
1.3.1. Creating a credential for Amazon Web Services 41
1.3.1.1. Prerequisites 42
1.3.1.2. Managing a credential by using the console 42

1.3.1.3. Creating an opaque secret by using the API 43
1.3.2. Creating a credential for Microsoft Azure 43
1.3.2.1. Prerequisites 43
1.3.2.2. Managing a credential by using the console 44
1.3.2.3. Creating an opaque secret by using the API 45
1.3.3. Creating a credential for Google Cloud Platform 45
1.3.3.1. Prerequisites 45
1.3.3.2. Managing a credential by using the console 46
1.3.3.3. Creating an opaque secret by using the API 47
1.3.4. Creating a credential for VMware vSphere 47
1.3.4.1. Prerequisites 47
1.3.4.2. Managing a credential by using the console 48
1.3.4.3. Creating an opaque secret by using the API 49
1.3.5. Creating a credential for Red Hat OpenStack 50
1.3.5.1. Prerequisites 50
1.3.5.2. Managing a credential by using the console 50
1.3.5.3. Creating an opaque secret by using the API 52
1.3.6. Creating a credential for Red Hat Virtualization 53
1.3.6.1. Prerequisites 53
1.3.6.2. Managing a credential by using the console 53
1.3.7. Creating a credential for Red Hat OpenShift Cluster Manager 54
1.3.7.1. Prerequisites 54
1.3.7.2. Managing a credential by using the console 54
1.3.8. Creating a credential for Ansible Automation Platform 55
1.3.8.1. Prerequisites 55
1.3.8.2. Managing a credential by using the console 55
1.3.9. Creating a credential for an on-premises environment 56
1.3.9.1. Prerequisites 56
1.3.9.2. Managing a credential by using the console 56
1.4. THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR CLUSTER LIFECYCLE OVERVIEW 56
1.4.1. Cluster lifecycle architecture 58
1.4.1.1. Hub cluster 59
1.4.1.2. Managed cluster 59
1.4.2. Release images 60
1.4.2.1. Creating a release image to deploy a cluster on a different architecture 62
1.4.2.2. Maintaining a custom list of release images while disconnected 64
1.4.3. Creating an infrastructure environment 65
1.4.3.1. Prerequisites 65
1.4.3.2. Enabling the Central Infrastructure Management service 66
1.4.3.2.1. Creating the AgentServiceConfig custom resource 67
1.4.3.2.2. Manually create the Provisioning custom resource (CR) 68
1.4.3.2.3. Enabling Central Infrastructure Management on Amazon Web Services 69
1.4.3.3. Creating your infrastructure environment with the console 70
1.4.4. Creating a cluster 71
1.4.4.1. Creating a cluster with the CLI 72
1.4.4.1.1. Prerequisites 72
1.4.4.1.2. Create a cluster with ClusterDeployment 72
1.4.4.1.3. Create a cluster with ClusterPool 72
1.4.4.2. Configuring additional manifests during cluster creation 72
1.4.4.3. Creating a cluster on Amazon Web Services 74
1.4.4.3.1. Prerequisites 74
1.4.4.3.2. Creating your cluster with the console 75
1.4.4.3.3. Adding your cluster to an existing cluster set 75

1.4.4.4. Creating a cluster on Microsoft Azure 77
1.4.4.4.1. Prerequisites 77
1.4.4.4.2. Creating your cluster with the console 77
1.4.4.4.3. Adding your cluster to an existing cluster set 78
1.4.4.5. Creating a cluster on Google Cloud Platform 79
1.4.4.5.1. Prerequisites 79
1.4.4.5.2. Creating your cluster with the console 80
1.4.4.5.3. Adding your cluster to an existing cluster set 80
1.4.4.6. Creating a cluster on VMware vSphere 81
1.4.4.6.1. Prerequisites 82
1.4.4.6.2. Creating your cluster with the console 82
1.4.4.6.3. Adding your cluster to an existing cluster set 82
1.4.4.7. Creating a cluster on Red Hat OpenStack Platform 84
1.4.4.7.1. Prerequisites 84
1.4.4.7.2. Creating your cluster with the console 85
1.4.4.7.3. Adding your cluster to an existing cluster set 85
1.4.4.8. Creating a cluster on Red Hat Virtualization 87
1.4.4.8.1. Prerequisites 88
1.4.4.8.2. Creating your cluster with the console 88
1.4.4.8.3. Adding your cluster to an existing cluster set 88
1.4.4.9. Creating a cluster in an on-premises environment 90
1.4.4.9.1. Prerequisites 90
1.4.4.9.2. Creating your cluster with the console 91
1.4.4.10. Creating a hosted cluster (Technology Preview) 92
1.4.4.10.1. Prerequisites 92
1.4.4.10.2. Creating a hosted cluster 93
1.4.4.11. Creating a cluster in a proxy environment 94
1.4.4.11.1. Enabling cluster-wide proxy on existing cluster add-ons 95
1.4.5. Hibernating a created cluster (Technology Preview) 96
1.4.5.1. Hibernate a cluster by using the console 96
1.4.5.2. Hibernate a cluster by using the CLI 97
1.4.5.3. Resuming normal operation of a hibernating cluster by using the console 97
1.4.5.4. Resuming normal operation of a hibernating cluster by using the CLI 97
1.4.6. Importing a target managed cluster to the hub cluster 98
1.4.6.1. Importing an existing cluster with the console 98
1.4.6.1.1. Prerequisites 98
1.4.6.1.2. Importing a cluster 99
1.4.6.1.2.1. Additional steps for running the import commands manually 100
1.4.6.1.2.2. Optional: Configure the cluster API address 102
1.4.6.1.3. Removing an imported cluster 102
1.4.6.2. Importing a managed cluster with the CLI 103
1.4.6.2.1. Prerequisites 103
1.4.6.2.2. Supported architectures 103
1.4.6.2.3. Preparing for import 103
1.4.6.2.4. Importing the cluster with the auto import secret 104
1.4.6.2.5. Importing the cluster with the manual commands 105
1.4.6.2.6. Importing the klusterlet add-on 106
1.4.6.2.7. Removing an imported cluster with the CLI 107
1.4.6.3. Importing a cluster with a custom ManagedClusterImageRegistry CRD 107
1.4.6.3.1. Importing a cluster with a ManagedClusterImageRegistry CRD 108
1.4.7. Removing a cluster from management 109
1.4.7.1. Removing a cluster by using the console 110
1.4.7.2. Removing a cluster by using the command line 110

1.4.7.3. Removing remaining resources after removing a cluster 111
1.4.7.4. Defragmenting the etcd database after removing a cluster 112
1.4.7.4.1. Prerequisites 112
1.4.7.4.2. Procedure 112
1.4.8. Scaling managed clusters 112
1.4.8.1. Scaling with MachinePool (Technology Preview) 112
1.4.8.1.1. Autoscaling 113
1.4.8.1.1.1. Enabling autoscaling 113
1.4.8.1.1.2. Disabling autoscaling 114
1.4.8.2. Scaling hosts to an infrastructure environment 115
1.4.8.2.1. Adding nodes to existing OpenShift Container Platform clusters deployed with CIM 116
1.4.9. Using cluster proxy add-ons 116
1.4.10. Configuring a specific cluster management role 117
1.4.11. Managing cluster labels 119
1.4.12. Configuring Ansible Tower tasks to run on managed clusters 119
1.4.12.1. Prerequisites 120
1.4.12.2. Configuring an AnsibleJob template to run on a cluster by using the console 120
1.4.12.3. Creating an AnsibleJob template 120
1.4.12.4. Viewing the status of an Ansible job 121
1.4.13. Creating and managing ManagedClusterSets 122
1.4.13.1. Global ManagedClusterSet 122
1.4.13.2. Creating a ManagedClusterSet 122
1.4.13.2.1. Creating a ManagedClusterSet by using the console 123
1.4.13.2.2. Creating a ManagedClusterSet by using the command line 123
1.4.13.3. Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet 123
1.4.13.3.1. Creating a ManagedClusterSetBinding resource 124
1.4.13.3.1.1. Creating a ManagedClusterSetBinding by using the console 124
1.4.13.3.1.2. Creating a ManagedClusterSetBinding by using the command line 125
1.4.13.4. Adding a cluster to a ManagedClusterSet 125
1.4.13.4.1. Adding clusters to a ManagedClusterSet by using the console 125
1.4.13.4.2. Adding clusters to a ManagedClusterSet by using the command line 126
1.4.13.5. Using ManagedClusterSets with Placement 127
1.4.13.5.1. Placement overview 127
1.4.13.5.2. Placement examples 129
1.4.13.5.3. Placement decision 131
1.4.13.5.4. Add-on status 132
1.4.13.5.5. Extensible scheduling 133
1.4.13.6. Using taints and tolerations to place managed clusters 134
1.4.13.6.1. Adding a taint to a managed cluster 134
1.4.13.6.2. Identifying built-in taints to reflect the status of managed clusters 134
1.4.13.6.3. Adding a toleration to a placement 135
1.4.13.6.4. Specifying a temporary toleration 136
1.4.13.7. Removing a managed cluster from a ManagedClusterSet 137
1.4.13.7.1. Removing a managed cluster from a ManagedClusterSet by using the console 137
1.4.13.7.2. Removing clusters from a ManagedClusterSet by using the command line 137
1.4.14. Managing cluster pools (Technology Preview) 138
1.4.14.1. Creating a cluster pool 138
1.4.14.1.1. Prerequisites 138
1.4.14.1.2. Create the cluster pool 139
1.4.14.2. Claiming clusters from cluster pools 140
1.4.14.2.1. Prerequisite 140
1.4.14.2.2. Claim the cluster from the cluster pool 140
1.4.14.3. Updating the cluster pool release image 140

1.4.14.4. Scaling cluster pools (Technology Preview) 141
1.4.14.5. Destroying a cluster pool 141
1.4.15. ClusterClaims 142
1.4.15.1. List existing ClusterClaims 144
1.4.15.2. Create custom ClusterClaims 144
1.4.16. Enabling ManagedServiceAccount add-ons (Technology Preview) 145
1.4.16.1. Prerequisites 145
1.4.16.2. Enabling ManagedServiceAccount 145
1.4.17. Upgrading your cluster 146
1.4.17.1. Selecting a channel 147
1.5. DISCOVERY SERVICE INTRODUCTION 147
1.5.1. Configure Discovery with the console 148
1.5.1.1. Prerequisites 148
1.5.1.2. Configure Discovery 148
1.5.1.3. View discovered clusters 148
1.5.1.4. Import discovered clusters 148
1.5.1.5. Prerequisites 149
1.5.1.6. Import Discovered clusters 149
1.5.2. Enable Discovery using the CLI 149
1.5.2.1. Prerequisites 149
1.5.2.2. Discovery set up and process 149
1.5.2.3. View discovered clusters 150
1.5.2.3.1. DiscoveredClusters 150
1.6. USING HOSTED CONTROL PLANE CLUSTERS (TECHNOLOGY PREVIEW) 150
1.6.1. Configuring hosted control planes 151
1.6.1.1. Configuring the hosting service cluster 151
1.6.1.1.1. Prerequisites 151
1.6.1.1.2. Configure the hosting service cluster 152
1.6.1.2. Deploying a hosted cluster 154
1.6.1.3. Deploying a customized hosted cluster 156
1.6.1.4. Accessing a hosting service cluster 158
1.6.2. Disabling the hosted control plane resources 158
1.6.2.1. Destroying a HyperShift hosted cluster 158
1.6.2.2. Uninstalling the HyperShift operator 158
1.7. APIS 158
1.7.1. Clusters API 159
1.7.1.1. Overview 159
1.7.1.1.1. URI scheme 159
1.7.1.1.2. Tags 159
1.7.1.2. Paths 159
1.7.1.2.1. Query all clusters 159
1.7.1.2.1.1. Description 159
1.7.1.2.1.2. Parameters 159
1.7.1.2.1.3. Responses 160
1.7.1.2.1.4. Consumes 160
1.7.1.2.1.5. Tags 160
1.7.1.2.2. Create a cluster 160
1.7.1.2.2.1. Description 160
1.7.1.2.2.2. Parameters 160
1.7.1.2.2.3. Responses 160
1.7.1.2.2.4. Consumes 161
1.7.1.2.2.5. Tags 161
1.7.1.2.2.6. Example HTTP request 161

1.7.1.2.2.6.1. Request body 161
1.7.1.2.3. Query a single cluster 161
1.7.1.2.3.1. Description 161
1.7.1.2.3.2. Parameters 162
1.7.1.2.3.3. Responses 162
1.7.1.2.3.4. Tags 162
1.7.1.2.4. Delete a cluster 162
1.7.1.2.4.1. Description 162
1.7.1.2.4.2. Parameters 162
1.7.1.2.4.3. Responses 163
1.7.1.2.4.4. Tags 163
1.7.1.3. Definitions 163
1.7.1.3.1. Cluster 163
1.7.2. Clustersets API (v1beta1) 164
1.7.2.1. Overview 164
1.7.2.1.1. URI scheme 164
1.7.2.1.2. Tags 164
1.7.2.2. Paths 164
1.7.2.2.1. Query all clustersets 164
1.7.2.2.1.1. Description 164
1.7.2.2.1.2. Parameters 165
1.7.2.2.1.3. Responses 165
1.7.2.2.1.4. Consumes 165
1.7.2.2.1.5. Tags 165
1.7.2.2.2. Create a clusterset 165
1.7.2.2.2.1. Description 165
1.7.2.2.2.2. Parameters 165
1.7.2.2.2.3. Responses 166
1.7.2.2.2.4. Consumes 166
1.7.2.2.2.5. Tags 166
1.7.2.2.2.6. Example HTTP request 166
1.7.2.2.2.6.1. Request body 166
1.7.2.2.3. Query a single clusterset 166
1.7.2.2.3.1. Description 166
1.7.2.2.3.2. Parameters 166
1.7.2.2.3.3. Responses 167
1.7.2.2.3.4. Tags 167
1.7.2.2.4. Delete a clusterset 167
1.7.2.2.4.1. Description 167
1.7.2.2.4.2. Parameters 167
1.7.2.2.4.3. Responses 168
1.7.2.2.4.4. Tags 168
1.7.2.3. Definitions 168
1.7.2.3.1. Clusterset 168
1.7.3. Clustersetbindings API (v1beta1) 168
1.7.3.1. Overview 168
1.7.3.1.1. URI scheme 169
1.7.3.1.2. Tags 169
1.7.3.2. Paths 169
1.7.3.2.1. Query all clustersetbindings 169
1.7.3.2.1.1. Description 169
1.7.3.2.1.2. Parameters 169
1.7.3.2.1.3. Responses 169

1.7.3.2.1.4. Consumes 169
1.7.3.2.1.5. Tags 170
1.7.3.2.2. Create a clustersetbinding 170
1.7.3.2.2.1. Description 170
1.7.3.2.2.2. Parameters 170
1.7.3.2.2.3. Responses 170
1.7.3.2.2.4. Consumes 170
1.7.3.2.2.5. Tags 170
1.7.3.2.2.6. Example HTTP request 171
1.7.3.2.2.6.1. Request body 171
1.7.3.2.3. Query a single clustersetbinding 171
1.7.3.2.3.1. Description 171
1.7.3.2.3.2. Parameters 171
1.7.3.2.3.3. Responses 171
1.7.3.2.3.4. Tags 172
1.7.3.2.4. Delete a clustersetbinding 172
1.7.3.2.4.1. Description 172
1.7.3.2.4.2. Parameters 172
1.7.3.2.4.3. Responses 172
1.7.3.2.4.4. Tags 173
1.7.3.3. Definitions 173
1.7.3.3.1. Clustersetbinding 173
1.7.4. Clusterview API (v1alpha1) 173
1.7.4.1. Overview 173
1.7.4.1.1. URI scheme 173
1.7.4.1.2. Tags 174
1.7.4.2. Paths 174
1.7.4.2.1. Get managed clusters 174
1.7.4.2.1.1. Description 174
1.7.4.2.1.2. Parameters 174
1.7.4.2.1.3. Responses 174
1.7.4.2.1.4. Consumes 174
1.7.4.2.1.5. Tags 174
1.7.4.2.2. List managed clusters 175
1.7.4.2.2.1. Description 175
1.7.4.2.2.2. Parameters 175
1.7.4.2.2.3. Responses 175
1.7.4.2.2.4. Consumes 175
1.7.4.2.2.5. Tags 175
1.7.4.2.2.6. Example HTTP request 175
1.7.4.2.2.6.1. Request body 175
1.7.4.2.3. Watch the managed cluster sets 176
1.7.4.2.3.1. Description 176
1.7.4.2.3.2. Parameters 176
1.7.4.2.3.3. Responses 176
1.7.4.2.4. List the managed cluster sets. 176
1.7.4.2.4.1. Description 176
1.7.4.2.4.2. Parameters 177
1.7.4.2.4.3. Responses 177
1.7.4.2.5. List the managed cluster sets. 177
1.7.4.2.5.1. Description 177
1.7.4.2.5.2. Parameters 177
1.7.4.2.5.3. Responses 177

1.7.4.2.6. Watch the managed cluster sets. 178
1.7.4.2.6.1. Description 178
1.7.4.2.6.2. Parameters 178
1.7.4.2.6.3. Responses 178
1.7.5. Managed service account (Technology Preview) 179
1.7.5.1. Overview 179
1.7.5.1.1. URI scheme 179
1.7.5.1.2. Tags 179
1.7.5.2. Paths 179
1.7.5.2.1. Create a ManagedServiceAccount 179
1.7.5.2.1.1. Description 179
1.7.5.2.1.2. Parameters 179
1.7.5.2.1.3. Responses 179
1.7.5.2.1.4. Consumes 180
1.7.5.2.1.5. Tags 180
1.7.5.2.1.5.1. Request body 180
1.7.5.2.2. Query a single ManagedServiceAccount 184
1.7.5.2.2.1. Description 184
1.7.5.2.2.2. Parameters 184
1.7.5.2.2.3. Responses 184
1.7.5.2.2.4. Tags 185
1.7.5.2.3. Delete a ManagedServiceAccount 185
1.7.5.2.3.1. Description 185
1.7.5.2.3.2. Parameters 185
1.7.5.2.3.3. Responses 185
1.7.5.2.3.4. Tags 185
1.7.5.3. Definitions 185
1.7.5.3.1. ManagedServiceAccount 185
1.7.6. MultiClusterEngine API 186
1.7.6.1. Overview 186
1.7.6.1.1. URI scheme 186
1.7.6.1.2. Tags 186
1.7.6.2. Paths 186
1.7.6.2.1. Create a MultiClusterEngine 186
1.7.6.2.1.1. Description 186
1.7.6.2.1.2. Parameters 186
1.7.6.2.1.3. Responses 187
1.7.6.2.1.4. Consumes 187
1.7.6.2.1.5. Tags 187
1.7.6.2.1.5.1. Request body 187
1.7.6.2.2. Query all MultiClusterEngines 191
1.7.6.2.2.1. Description 191
1.7.6.2.2.2. Parameters 191
1.7.6.2.2.3. Responses 192
1.7.6.2.2.4. Consumes 192
1.7.6.2.2.5. Tags 192
1.7.6.2.3. Delete a MultiClusterEngine operator 192
1.7.6.2.3.1. Parameters 192
1.7.6.2.3.2. Responses 192
1.7.6.2.3.3. Tags 193
1.7.6.3. Definitions 193
1.7.6.3.1. MultiClusterEngine 193
1.7.6.3.2. List of specs 193

1.7.7. Placements API (v1beta1) 194
1.7.7.1. Overview 194
1.7.7.1.1. URI scheme 194
1.7.7.1.2. Tags 194
1.7.7.2. Paths 194
1.7.7.2.1. Query all Placements 194
1.7.7.2.1.1. Description 194
1.7.7.2.1.2. Parameters 194
1.7.7.2.1.3. Responses 194
1.7.7.2.1.4. Consumes 195
1.7.7.2.1.5. Tags 195
1.7.7.2.2. Create a Placement 195
1.7.7.2.2.1. Description 195
1.7.7.2.2.2. Parameters 195
1.7.7.2.2.3. Responses 195
1.7.7.2.2.4. Consumes 196
1.7.7.2.2.5. Tags 196
1.7.7.2.2.6. Example HTTP request 196
1.7.7.2.2.6.1. Request body 196
1.7.7.2.3. Query a single Placement 196
1.7.7.2.3.1. Description 196
1.7.7.2.3.2. Parameters 197
1.7.7.2.3.3. Responses 197
1.7.7.2.3.4. Tags 197
1.7.7.2.4. Delete a Placement 197
1.7.7.2.4.1. Description 197
1.7.7.2.4.2. Parameters 197
1.7.7.2.4.3. Responses 198
1.7.7.2.4.4. Tags 198
1.7.7.3. Definitions 198
1.7.7.3.1. Placement 198
1.7.8. PlacementDecisions API (v1beta1) 200
1.7.8.1. Overview 200
1.7.8.1.1. URI scheme 200
1.7.8.1.2. Tags 200
1.7.8.2. Paths 200
1.7.8.2.1. Query all PlacementDecisions 200
1.7.8.2.1.1. Description 200
1.7.8.2.1.2. Parameters 200
1.7.8.2.1.3. Responses 200
1.7.8.2.1.4. Consumes 201
1.7.8.2.1.5. Tags 201
1.7.8.2.2. Create a PlacementDecision 201
1.7.8.2.2.1. Description 201
1.7.8.2.2.2. Parameters 201
1.7.8.2.2.3. Responses 201
1.7.8.2.2.4. Consumes 201
1.7.8.2.2.5. Tags 201
1.7.8.2.2.6. Example HTTP request 202
1.7.8.2.2.6.1. Request body 202
1.7.8.2.3. Query a single PlacementDecision 202
1.7.8.2.3.1. Description 202
1.7.8.2.3.2. Parameters 202

1.7.8.2.3.3. Responses 202
1.7.8.2.3.4. Tags 203
1.7.8.2.4. Delete a PlacementDecision 203
1.7.8.2.4.1. Description 203
1.7.8.2.4.2. Parameters 203
1.7.8.2.4.3. Responses 203
1.7.8.2.4.4. Tags 203
1.7.8.3. Definitions 204
1.7.8.3.1. PlacementDecision 204
1.8. TROUBLESHOOTING 204
1.8.1. Documented troubleshooting 204
1.8.2. Running the must-gather command to troubleshoot 205
1.8.2.1. Must-gather scenarios 205
1.8.2.2. Must-gather procedure 205
1.8.2.3. Must-gather in a disconnected environment 206
1.8.3. Troubleshooting installation status stuck in installing or pending 206
1.8.3.1. Symptom: Stuck in Pending status 206
1.8.3.2. Resolving the problem: Adjust worker node sizing 206
1.8.4. Troubleshooting reinstallation failure 206
1.8.4.1. Symptom: Reinstallation failure 207
1.8.4.2. Resolving the problem: Reinstallation failure 207
1.8.5. Troubleshooting an offline cluster 207
1.8.5.1. Symptom: Cluster status is offline 207
1.8.5.2. Resolving the problem: Cluster status is offline 207
1.8.6. Troubleshooting a managed cluster import failure 208
1.8.6.1. Symptom: Imported cluster not available 208
1.8.6.2. Resolving the problem: Imported cluster not available 208
1.8.7. Reimporting cluster fails with unknown authority error 209
1.8.7.1. Symptom: Reimporting cluster fails with unknown authority error 209
1.8.7.2. Identifying the problem: Reimporting cluster fails with unknown authority error 209
1.8.7.3. Resolving the problem: Reimporting cluster fails with unknown authority error 210
1.8.8. Troubleshooting cluster with pending import status 210
1.8.8.1. Symptom: Cluster with pending import status 211
1.8.8.2. Identifying the problem: Cluster with pending import status 211
1.8.8.3. Resolving the problem: Cluster with pending import status 211
1.8.9. Troubleshooting imported clusters offline after certificate change 211
1.8.9.1. Symptom: Clusters offline after certificate change 211
1.8.9.2. Identifying the problem: Clusters offline after certificate change 211
1.8.9.3. Resolving the problem: Clusters offline after certificate change 212
1.8.10. Troubleshooting cluster status changing from offline to available 213
1.8.10.1. Symptom: Cluster status changing from offline to available 213
1.8.10.2. Resolving the problem: Cluster status changing from offline to available 213
1.8.11. Troubleshooting cluster creation on VMware vSphere 213
1.8.11.1. Managed cluster creation fails with certificate IP SAN error 214
1.8.11.1.1. Symptom: Managed cluster creation fails with certificate IP SAN error 214
1.8.11.1.2. Identifying the problem: Managed cluster creation fails with certificate IP SAN error 214
1.8.11.1.3. Resolving the problem: Managed cluster creation fails with certificate IP SAN error 214
1.8.11.2. Managed cluster creation fails with unknown certificate authority 214
1.8.11.2.1. Symptom: Managed cluster creation fails with unknown certificate authority 214
1.8.11.2.2. Identifying the problem: Managed cluster creation fails with unknown certificate authority 214
1.8.11.2.3. Resolving the problem: Managed cluster creation fails with unknown certificate authority 214
1.8.11.3. Managed cluster creation fails with expired certificate 214
1.8.11.3.1. Symptom: Managed cluster creation fails with expired certificate 214

1.8.11.3.2. Identifying the problem: Managed cluster creation fails with expired certificate 215
1.8.11.3.3. Resolving the problem: Managed cluster creation fails with expired certificate 215
1.8.11.4. Managed cluster creation fails with insufficient privilege for tagging 215
1.8.11.4.1. Symptom: Managed cluster creation fails with insufficient privilege for tagging 215
1.8.11.4.2. Identifying the problem: Managed cluster creation fails with insufficient privilege for tagging 215
1.8.11.4.3. Resolving the problem: Managed cluster creation fails with insufficient privilege for tagging 215
1.8.11.5. Managed cluster creation fails with invalid dnsVIP 215
1.8.11.5.1. Symptom: Managed cluster creation fails with invalid dnsVIP 215
1.8.11.5.2. Identifying the problem: Managed cluster creation fails with invalid dnsVIP 215
1.8.11.5.3. Resolving the problem: Managed cluster creation fails with invalid dnsVIP 216
1.8.11.6. Managed cluster creation fails with incorrect network type 216
1.8.11.6.1. Symptom: Managed cluster creation fails with incorrect network type 216
1.8.11.6.2. Identifying the problem: Managed cluster creation fails with incorrect network type 216
1.8.11.6.3. Resolving the problem: Managed cluster creation fails with incorrect network type 216
1.8.11.7. Managed cluster creation fails with an error processing disk changes 216
1.8.11.7.1. Symptom: Adding the VMware vSphere managed cluster fails due to an error processing disk changes 216
1.8.11.7.2. Identifying the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes 216
1.8.11.7.3. Resolving the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes 217
1.8.12. Troubleshooting cluster in console with pending or failed status 217
1.8.12.1. Symptom: Cluster in console with pending or failed status 217
1.8.12.2. Identifying the problem: Cluster in console with pending or failed status 217
1.8.12.3. Resolving the problem: Cluster in console with pending or failed status 218
1.8.13. Troubleshooting OpenShift Container Platform version 3.11 cluster import failure 218
1.8.13.1. Symptom: OpenShift Container Platform version 3.11 cluster import failure 218
1.8.13.2. Identifying the problem: OpenShift Container Platform version 3.11 cluster import failure 218
1.8.13.3. Resolving the problem: OpenShift Container Platform version 3.11 cluster import failure 219
1.8.14. Troubleshooting Klusterlet with degraded conditions 219
1.8.14.1. Symptom: Klusterlet is in the degraded condition 219
1.8.14.2. Identifying the problem: Klusterlet is in the degraded condition 219
1.8.14.3. Resolving the problem: Klusterlet is in the degraded condition 219
1.8.15. Namespace remains after deleting a cluster 220
1.8.15.1. Symptom: Namespace remains after deleting a cluster 220
1.8.15.2. Resolving the problem: Namespace remains after deleting a cluster 220
1.8.16. Auto-import-secret-exists error when importing a cluster 221
1.8.16.1. Symptom: Auto import secret exists error when importing a cluster 221
1.8.16.2. Resolving the problem: Auto-import-secret-exists error when importing a cluster 221

CHAPTER 1. THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR OVERVIEW

- Console overview
  - Requirements and recommendations
  - Networking
  - Console overview
  - Role-based access control
- Install
  - Installing while connected online
  - Installing on disconnected networks
  - MultiClusterEngine advanced configuration
  - Uninstalling
- Managing credentials
  - Creating a credential for Amazon Web Services
  - Creating a credential for Microsoft Azure
  - Creating a credential for Google Cloud Platform
  - Creating a credential for VMware vSphere
  - Creating a credential for Red Hat OpenStack Platform
  - Creating a credential for Red Hat Virtualization
  - Creating a credential for Red Hat OpenShift Cluster Manager
  - Creating a credential for Ansible Automation Platform
  - Creating a credential for an on-premises environment
- Cluster lifecycle architecture
- Managing credentials overview
- Release images
  - Maintaining a custom list of release images while disconnected
- Creating a cluster
  - Creating a cluster with the CLI
  - Configuring additional manifests during cluster creation
  - Creating a cluster on Amazon Web Services

    - Creating a cluster on Microsoft Azure
    - Creating a cluster on Google Cloud Platform
    - Creating a cluster on VMware vSphere
    - Creating a cluster on Red Hat OpenStack Platform
    - Creating a cluster on Red Hat Virtualization
    - Creating a cluster in an on-premise environment
    - Creating a cluster in a proxy environment
    - Creating an infrastructure environment
- Importing a cluster
    - Importing a target managed cluster to the hub cluster
    - Importing an existing cluster with the console
    - Importing a managed cluster with the CLI
    - Importing a cluster with a custom ManagedClusterImageRegistry CRD
- Hibernating a created cluster (Technology Preview)
- Scaling managed clusters
    - Scaling with MachinePool (Technology Preview)
    - Scaling cluster pools
    - Scaling hosts to an infrastructure environment
- Using taints and tolerations to place managed clusters
- Creating a cluster in a proxy environment
- Enabling cluster proxy add-ons
- Configuring a specific cluster management role
- Managing cluster labels
- Configuring Ansible Tower tasks to run on managed clusters
- Creating and managing ManagedClusterSets
    - Creating a ManagedClusterSet
    - Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet
    - Creating a ManagedClusterSetBinding resource
    - Adding a cluster to a ManagedClusterSet

    - Using ManagedClusterSets with Placement
    - Removing a cluster from a ManagedClusterSet
- Managing cluster pools (Technology Preview)
    - Creating a cluster pool
    - Claiming clusters from cluster pools
    - Updating the cluster pool release image
    - Destroying a cluster pool
- ClusterClaims
    - List existing ClusterClaims
    - Create custom ClusterClaims
- Enabling ManagedServiceAccount
- Upgrading your cluster
- Removing a cluster from management
- Discovery service introduction
    - Configure Discovery with the console
    - Configure Discovery using the CLI
- Using hosted control plane clusters (Technology Preview)
    - Configuring hosted control planes
    - Disabling hosted control plane resources
- APIs
    - Clusters API
    - ClusterSets API (v1beta1)
    - ClusterSetBindings API (v1beta1)
    - Clusterview API
    - Managed service account (Technology Preview)
    - MultiClusterEngine API
    - Placements API (v1alpha1)
    - PlacementDecisions API (v1alpha1)
- Troubleshooting
    - Running the must gather command to troubleshoot

- Troubleshooting installation status stuck in installing or pending
- Troubleshooting reinstallation failure
- Troubleshooting an offline cluster
- Troubleshooting a managed cluster import failure
- Troubleshooting cluster with pending import status
- Troubleshooting imported clusters offline after certificate change
- Troubleshooting cluster status changing from offline to available
- Troubleshooting cluster creation on VMware vSphere
- Troubleshooting cluster in console with pending or failed status
- Troubleshooting OpenShift Container Platform version 3.11 cluster import failure
- Troubleshooting Klusterlet with degraded conditions
- Namespace remains after deleting a cluster
- Auto-import-secret-exists error when importing a cluster

1.1. ABOUT THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR

The multicluster engine for Kubernetes operator is a cluster lifecycle operator that provides multicluster lifecycle capabilities for Red Hat OpenShift Container Platform. See The multicluster engine for Kubernetes operator 2.1 support matrix, as well as the following documentation about the multicluster engine for Kubernetes operator:

- Requirements and recommendations
- Console overview
- Role-based access control

1.1.1. Requirements and recommendations

Before you install the multicluster engine for Kubernetes operator, review the following system configuration requirements and settings:

- Supported browsers and platforms
- Networking

Important: You must install multicluster engine for Kubernetes operator on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. If you are using Red Hat Advanced Cluster Management version 2.5 or later, then multicluster engine for Kubernetes is already installed on the cluster with it.

1.1.1.1. Supported browsers and platforms

See important information about supported browsers and features in the multicluster engine for Kubernetes operator 2.1 support matrix.

1.1.2. Networking

Learn about network requirements for the multicluster engine for Kubernetes operator hub cluster and managed clusters.

Important: The trusted CA bundle is available in the multicluster engine for Kubernetes operator namespace, but that enhancement requires changes to your network. The trusted CA bundle ConfigMap uses the default name of **trusted-ca-bundle**. You can change this name by providing it to the operator in an environment variable named **TRUSTED_CA_BUNDLE**. See Configuring the cluster-wide proxy in the *Networking* section of Red Hat OpenShift Container Platform for more information. on.

- Hub cluster network configuration
- Managed cluster network configuration
- Additional networking requirements when installing using the infrastructure operator
- Additional networking requirements when installing using the Hive Operator
- Hosted control planes networking requirements (Technology Preview)

1.1.2.1. Hub cluster network configuration

You can refer to the configuration for your hub cluster network.

See the hub cluster network requirements in the following table:

Direction	Connection	Port (if specified)
Outbound	API of the cloud provider	
Outbound	(Optional) Kubernetes API server of the provisioned managed cluster	6443
Outbound and inbound	The **WorkManager** service route on the managed cluster	443
Inbound	The Kubernetes API server of the multicluster engine for Kubernetes cluster from the managed cluster	6443

1.1.2.2. Managed cluster network configuration

For the managed cluster networking requirements, see the following table:

Direction	Connection	Port (if specified)
Outbound and inbound	Kubernetes API server of the multicluster engine for Kubernetes cluster	6443

1.1.2.3. Additional networking requirements when installing using the infrastructure operator

When you are installing bare metal managed clusters with the Infrastructure Operator, see the following table for the additional networking requirements:

Direction	Protocol	Connection	Port (if specified)
Hub cluster outbound to the ISO/rootfs image repository	HTTPS (HTTP in a disconnected environment	Used to create an ISO image on the Red Hat Advanced Cluster Management hub	443 (80 in disconnected environments)
Hub cluster outbound to BMC interface at single node OpenShift Container Platform managed cluster	HTTPS (HTTP in disconnected environment)	Boot the OpenShift Container Platform cluster	443
Outbound from the OpenShift Container Platform managed cluster to the hub cluster	HTTPS	Reports hardware information using the **assistedService** route	443
Outbound from the OpenShift Container Platform managed cluster to the ISO/rootfs image repository	HTTP, HTTPS or TLS	Downloads the rootfs image	HTTP 80, HTTPS 443
Outbound from the OpenShift Container Platform managed cluster to the **registry.redhat.com** image repository	HTTPS/TLS	Downloads the image	443

1.1.2.4. Additional networking requirements when installing using the Hive Operator

When you are installing bare metal managed clusters with the Hive Operator, which includes using Central Infrastructure Management, you must configure a layer 2 or layer 3 port connection between the hub cluster and the **libvirt** provisioning host. This connection to the provisioning host is required during the creation of a base metal cluster with Hive. See the following table for more information:

Direction	Protocol	Connection	Port (if specified)
Hub cluster outbound and inbound to the **libvirt** provisioning host	IP	Connects the hub cluster, where the Hive operator is installed, to the **libvirt** provisioning host that serves as a bootstrap when creating the bare metal cluster	

Note: These requirements only apply when installing, and are not required when upgrading clusters that were installed with Infrastructure Operator.

1.1.2.4.1. Hosted control planes networking requirements (Technology Preview)

When you use hosted control planes, the **HypershiftDeployment** resource must have connectivity to the endpoints listed in the following table:

Direction	Connection	Port (if specified)
Outbound	OpenShift Container Platform control-plane and worker nodes	
Outbound	For hosted clusters on Amazon Web Services only: Outbound connection to AWS API and S3 API	
Outbound	For hosted clusters on Microsoft Azure cloud services only: Outbound connection to Azure API	
Outbound	OpenShift Container Platform image repositories that store the ISO images of the coreOS and the image registry for OpenShift Container Platform pods	

1.1.3. Console overview

OpenShift Container Platform console plug-ins are available with OpenShift Container Platform 4.10 web console and can be integrated. To use this feature, the console plug-ins must remain enabled. The multicluster engine for Kubernetes operator displays certain console features from **Infrastructure** and **Credentials** navigation items. If you install Red Hat Advanced Cluster Management, you see more console capability.

Note: For OpenShift Container Platform 4.10 with the plug-ins enabled, you can access Red Hat Advanced Cluster Management within the OpenShift Container Platform console from the cluster switcher by selecting **All Clusters** from the drop-down menu.

1. To disable the plug-in, be sure you are in the *Administrator* perspective in the OpenShift Container Platform console.
2. Find **Administration** in the navigation and click **Cluster Settings**, then click *Configuration* tab.
3. From the list of *Configuration resources*, click the **Console** resource with the **operator.openshift.io** API group, which contains cluster-wide configuration for the web console.
4. Click on the *Console plug-ins* tab. The **mce** plug-in is listed. **Note:** If Red Hat Advanced Cluster Management is installed, it is also listed as **acm**.
5. Modify plug-in status from the table. In a few moments, you are prompted to refresh the console.

1.1.4. Role-based access control

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

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

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

- Overview of roles
- Cluster lifecycle RBAC

1.1.4.1. Overview of roles

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

1.1.4.1.1. Table of role definition

Role	Definition
cluster-admin	This is an OpenShift Container Platform default role. A user with cluster binding to the **cluster-admin** role is an OpenShift Container Platform super user, who has all access.
open-cluster-management:cluster-manager-admin	A user with cluster binding to the **open-cluster-management:cluster-manager-admin** role is a super user, who has all access. This role allows the user to create a **ManagedCluster** resource.

Role	Definition
open-cluster-management:admin:<managed_cluster_name>	A user with cluster binding to the **open-cluster-management:admin:<managed_cluster_name>** role has administrator access to the **ManagedCluster** resource named, **<managed_cluster_name>**. When a user has a managed cluster, this role is automatically created.
open-cluster-management:view:<managed_cluster_name>	A user with cluster binding to the **open-cluster-management:view:<managed_cluster_name>** role has view access to the **ManagedCluster** resource named, **<managed_cluster_name>**.
open-cluster-management:managedclusterset:admin:<managed_clusterset_name>	A user with cluster binding to the **open-cluster-management:managedclusterset:admin:<managed_clusterset_name>** role has administrator access to **ManagedCluster** resource named **<managed_clusterset_name>**. The user also has administrator access to **managedcluster.cluster.open-cluster-management.io**, **clusterclaim.hive.openshift.io**, **clusterdeployment.hive.openshift.io**, and **clusterpool.hive.openshift.io** resources, which has the managed cluster set labels: **cluster.open-cluster-management.io** and **clusterset=<managed_clusterset_name>**. A role binding is automatically generated when you are using a cluster set. See Creating and managing ManagedClusterSets to learn how to manage the resource.
open-cluster-management:managedclusterset:view:<managed_clusterset_name>	A user with cluster binding to the **open-cluster-management:managedclusterset:view:<managed_clusterset_name>** role has view access to the **ManagedCluster** resource named, <managed_clusterset_name>`. The user also has view access to **managedcluster.cluster.open-cluster-management.io**, **clusterclaim.hive.openshift.io**, **clusterdeployment.hive.openshift.io**, and **clusterpool.hive.openshift.io** resources, which has the managed cluster set labels: **cluster.open-cluster-management.io**, **clusterset=<managed_clusterset_name>**. For more details on how to manage managed cluster set resources, see Creating and managing ManagedClusterSets.

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

Important:

- Any user can create projects from OpenShift Container Platform, which gives administrator role permissions for the namespace.
- If a user does not have role access to a cluster, the cluster name is not visible. The cluster name is displayed with the following symbol: **-**.

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

1.1.4.1.2. Cluster lifecycle RBAC

- To create and administer all managed clusters, see the following information:
  - Create a cluster role binding to the cluster role **open-cluster-management:cluster-manager-admin** by entering the following command:

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

    This role is a super user, which has access to all resources and actions. You can create cluster-scoped **managedcluster** resources, the namespace for the resources that manage the managed cluster, and the resources in the namespace with this role. You can also access provider connections and bare metal assets that are used to create managed clusters with this role.
- To administer a managed cluster named **cluster-name**, see the following:
  - Create a cluster role binding to the cluster role **open-cluster-management:admin:<cluster-name>** by entering the following command:

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

    This role has read and write access to the cluster-scoped **managedcluster** resource. This is needed because the **managedcluster** is a cluster-scoped resource and not a namespace-scoped resource.
  - Create a namespace role binding to the cluster role **admin** by entering the following command:

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

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

- To view a managed cluster named **cluster-name**, see the following:
  - Create a cluster role binding to the cluster role **open-cluster-management:view:<cluster-name>** by entering the following command:

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

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

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

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

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

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

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

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

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

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

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

1.1.4.1.2.1. Cluster pools RBAC

View the following cluster pool RBAC operations.

- To use cluster pool provision clusters:
  - As a cluster administrator, create a managed cluster set and grant administrator permission to roles by adding the role to the group.
    - Grant **admin** permission to the **server-foundation-clusterset** managed cluster set with the following command:

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

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

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

- Create a namespace for the cluster pool, **server-foundation-clusterpool**.
  - Grant **admin** permission to **server-foundation-clusterpool** for the **server-foundation-team-admin** by running the following commands:

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

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

- As a team administrator, create a cluster pool named **ocp46-aws-clusterpool** with a cluster set label, **cluster.open-cluster-management.io/clusterset=server-foundation-clusterset** in the cluster pool namespace.
  - The **server-foundation-webhook** checks if the cluster pool has the cluster set label, and if the user has permission to create cluster pools in the cluster set.
  - The **server-foundation-controller** grants **view** permission to the **server-foundation-clusterpool** namespace for **server-foundation-team-user**.
- When a cluster pool is created, the cluster pool creates a **clusterdeployment**.
  - The **server-foundation-controller** grants **admin** permission to the **clusterdeployment** namespace for **server-foundation-team-admin**.
  - The **server-foundation-controller** grants **view** permission **clusterdeployment** namespace for **server-foundation-team-user**.
    Note: As a **team-admin** and **team-user**, you have **admin** permission to the **clusterpool**, **clusterdeplyment**, and **clusterclaim**.

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

1.1.4.1.2.2. Console RBAC table for cluster lifecycle

Resource	Admin	Edit	View
Clusters	read, update, delete	no data	read
Cluster sets	get, update, bind, join	edit role not mentioned	get
Managed clusters	read, update, delete	no edit role mentioned	get
Provider connections	create, read, update, and delete	no data	read
Bare metal asset	create, read, update, delete	no data	read

1.1.4.1.2.3. Table of RBAC API table for cluster lifecycle

API	Admin	Edit	View
managedclusters.cluster.open-cluster-management.io (You can use **mcl** (singular) or **mcls** (plural) in commands for this API.)	create, read, update, delete	read, update	read
managedclusters.view.open-cluster-management.io (You can use **mcv** (singular) or **mcvs** (plural) in commands for this API.)	read	read	read
managedclusters.register.open-cluster-management.io/accept	update	update	none
managedclusterset.cluster.open-cluster-management.io (You can use **mclset** (singular) or **mclsets** (plural) in commands for this API.)	create, read, update, delete	read, update	read
managedclustersets.view.open-cluster-management.io	read	read	read
managedclustersetbinding.cluster.open-cluster-management.io (You can use **mclsetbinding** (singular) or **mclsetbindings** (plural) in commands for this API.)	create, read, update, delete	read, update	read
baremetalassets.inventory.open-cluster-management.io	create, read, update, delete	read, update	read
klusterletaddonconfigs.agent.open-cluster-management.io	create, read, update, delete	read, update	read

API	Admin	Edit	View
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
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

1.1.4.1.2.4. Credentials role-based access control

The access to credentials is controlled by Kubernetes. Credentials are stored and secured as Kubernetes secrets. The following permissions apply to accessing secrets:

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

1.2. INSTALL

The multicluster engine for Kubernetes operator is a software operator that enhances cluster fleet management. The multicluster engine for Kubernetes operator supportsRed Hat OpenShift Container Platform and Kubernetes cluster lifecycle management across clouds and data centers.

See the following documentation:

- Installing while connected online

- Installing on disconnected networks
- Uninstalling
- MultiClusterEngine advanced configuration

1.2.1. Installing while connected online

The multicluster engine for Kubernetes operator is installed with Operator Lifecycle Manager, which manages the installation, upgrade, and removal of the components that encompass the multicluster engine for Kubernetes operator.

Required access: Cluster administrator

Important:

- You must install the multicluster engine for Kubernetes operator on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. The multicluster engine for Kubernetes operator cannot co-exist with Red Hat Advanced Cluster Management for Kubernetes on versions earlier than 2.5 because they provide some of the same management components. It is recommended that you install multicluster engine for Kubernetes operator on a cluster that has never previously installed Red Hat Advanced Cluster Management. If you are using Red Hat Advanced Cluster Management for Kubernetes at version 2.5 or later, then multicluster engine for Kubernetes operator is already installed on the cluster with it.
- For OpenShift Container Platform Dedicated environment, you must have **cluster-admin** permissions. By default **dedicated-admin** role does not have the required permissions to create namespaces in the OpenShift Container Platform Dedicated environment.
- By default, the multicluster engine for Kubernetes operator components are installed on worker nodes of your OpenShift Container Platform cluster without any additional configuration. You can install multicluster engine for Kubernetes operator onto worker nodes by using the OpenShift Container Platform OperatorHub web console interface, or by using the OpenShift Container Platform CLI.
- If you have configured your OpenShift Container Platform cluster with infrastructure nodes, you can install multicluster engine for Kubernetes operator onto those infrastructure nodes by using the OpenShift Container Platform CLI with additional resource parameters. Not all of the multicluster engine for Kubernetes operator components have infrastructure node support, so some worker nodes are still required when installing multicluster engine for Kubernetes operator on infrastructure nodes. See the Installing multicluster engine on infrastructure nodes section for those details.
- If you plan to import Kubernetes clusters that were not created by OpenShift Container Platform or multicluster engine for Kubernetes, you will need to configure an image pull secret. For information on how to configure an image pull secret and other advanced configurations, see options in the Advanced configuration section of this documentation.
    - Prerequisites
    - Confirm your OpenShift Container Platform installation
    - Installing from the OperatorHub web console interface
    - Installing from the OpenShift Container Platform CLI

- Installing multicluster engine on infrastructure nodes

1.2.1.1. Prerequisites

Before you install multicluster engine for Kubernetes, see the following requirements:

- Your RedHat OpenShift Container Platform cluster must have access to the multicluster engine for Kubernetes operator in the OperatorHub catalog from the OpenShift Container Platform console.
- You need access to the catalog.redhat.com.
- OpenShift Container Platform version 4.8, or later, must be deployed in your environment, and you must be logged into with the OpenShift Container Platform CLI. See the following install documentation for OpenShift Container Platform:
  - OpenShift Container Platform version 4.8
  - OpenShift Container Platform version 4.9
  - OpenShift Container Platform version 4.10
  - OpenShift Container Platform version 4.11
- Your OpenShift Container Platform command line interface (CLI) must be configured to run **oc** commands. See Getting started with the CLI for information about installing and configuring the OpenShift Container Platform CLI.
- Your OpenShift Container Platform permissions must allow you to create a namespace.
- You must have an Internet connection to access the dependencies for the operator.
- To install in a OpenShift Container Platform Dedicated environment, see the following:
  - You must have the OpenShift Container Platform Dedicated environment configured and running.
  - You must have **cluster-admin** authority to the OpenShift Container Platform Dedicated environment where you are installing the engine.
- If you plan to create managed clusters by using the Assisted Installer that is provided with Red Hat OpenShift Container Platform, see Preparing to install with the Assisted Installer topic in the OpenShift Container Platform documentation for the requirements.

1.2.1.2. Confirm your OpenShift Container Platform installation

You must have a supported OpenShift Container Platform version, including the registry and storage services, installed and working. For more information about installing OpenShift Container Platform, see the OpenShift Container Platform documentation.

1. Verify that multicluster engine for Kubernetes operator is not already installed on your OpenShift Container Platform cluster. The multicluster engine for Kubernetes operator allows only one single installation on each OpenShift Container Platform cluster. Continue with the following steps if there is no installation.
2. To ensure that the OpenShift Container Platform cluster is set up correctly, access the OpenShift Container Platform web console with the following command:

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

See the following example output:

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

3. Open the URL in your browser and check the result. If the console URL displays **console-openshift-console.router.default.svc.cluster.local**, set the value for **openshift_master_default_subdomain** when you install OpenShift Container Platform. See the following example of a URL: **https://console-openshift-console.apps.new-coral.purple-chesterfield.com**.

You can proceed to install multicluster engine for Kubernetes operator.

1.2.1.3. Installing from the OperatorHub web console interface

Best practice: From the *Administrator* view in your OpenShift Container Platform navigation, install the OperatorHub web console interface that is provided with OpenShift Container Platform.

1. Select **Operators** > **OperatorHub** to access the list of available operators, and select *multicluster engine for Kubernetes* operator.
2. Click **Install**.
3. On the *Operator Installation* page, select the options for your installation:
   - Namespace:
     - The multicluster engine for Kubernetes operator engine must be installed in its own namespace, or project.
     - By default, the OperatorHub console installation process creates a namespace titled **multicluster-engine**. **Best practice:** Continue to use the **multicluster-engine** namespace if it is available.
     - If there is already a namespace named **multicluster-engine**, select a different namespace.
   - Channel: The channel that you select corresponds to the release that you are installing. When you select the channel, it installs the identified release, and establishes that the future errata updates within that release are obtained.
   - Approval strategy: The approval strategy identifies the human interaction that is required for applying updates to the channel or release to which you subscribed.
     - Select **Automatic**, which is selected by default, to ensure any updates within that release are automatically applied.
     - Select **Manual** to receive a notification when an update is available. If you have concerns about when the updates are applied, this might be best practice for you.

   Note: To upgrade to the next minor release, you must return to the *OperatorHub* page and select a new channel for the more current release.
4. Select **Install** to apply your changes and create the operator.

5. See the following process to create the *MultiClusterEngine* custom resource.

   a. In the OpenShift Container Platform console navigation, select **Installed Operators** > **multicluster engine for Kubernetes**.

   b. Select the **MultiCluster Engine** tab.

   c. Select **Create MultiClusterEngine**.

   d. Update the default values in the YAML file. See options in the *MultiClusterEngine advanced configuration* section of the documentation.

      - The following example shows the default template that you can copy into the editor:

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

6. Select **Create** to initialize the custom resource. It can take up to 10 minutes for the multicluster engine for Kubernetes operator engine to build and start.
   After the *MultiClusterEngine* resource is created, the status for the resource is **Available** on the *MultiCluster Engine* tab.

1.2.1.4. Installing from the OpenShift Container Platform CLI

1. Create a multicluster engine for Kubernetes operator engine namespace where the operator requirements are contained. Run the following command, where **namespace** is the name for your multicluster engine for Kubernetes engine namespace. The value for **namespace** might be referred to as *Project* in the OpenShift Container Platform environment:

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

2. Switch your project namespace to the one that you created. Replace **namespace** with the name of the multicluster engine for Kubernetes engine namespace that you created in step 1.

```
oc project <namespace>
```

3. Create a YAML file to configure an **OperatorGroup** resource. Each namespace can have only one operator group. Replace **default** with the name of your operator group. Replace **namespace** with the name of your project namespace. See the following example:

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

4. Run the following command to create the **OperatorGroup** resource. Replace **operator-group** with the name of the operator group YAML file that you created:

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

5. Create a YAML file to configure an OpenShift Container Platform Subscription. Your file should look similar to the following example:

```
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
```

Note: For installing the multicluster engine for Kubernetes engine on infrastructure nodes, the see Operator Lifecycle Manager Subscription additional configuration section.

6. Run the following command to create the OpenShift Container Platform Subscription. Replace **subscription** with the name of the subscription file that you created:

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

7. Create a YAML file to configure the **MultiClusterEngine** custom resource. Your default template should look similar to the following example:

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

Note: For installing the multicluster engine for Kubernetes operator on infrastructure nodes, see the MultiClusterEngine custom resource additional configuration section:

8. Run the following command to create the **MultiClusterEngine** custom resource. Replace **custom-resource** with the name of your custom resource file:

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

If this step fails with the following error, the resources are still being created and applied. Run the command again in a few minutes when the resources are created:

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

9. Run the following command to get the custom resource. It can take up to 10 minutes for the **MultiClusterEngine** custom resource status to display as **Available** in the **status.phase** field after you run the following command:

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

If you are reinstalling the multicluster engine for Kubernetes operator and the pods do not start, see Troubleshooting reinstallation failure for steps to work around this problem.

Notes:

- A **ServiceAccount** with a **ClusterRoleBinding** automatically gives cluster administrator privileges to multicluster engine for Kubernetes operator and to any user credentials with access to the namespace where you install multicluster engine for Kubernetes operator.

1.2.1.5. Installing on infrastructure nodes

An OpenShift Container Platform cluster can be configured to contain infrastructure nodes for running approved management components. Running components on infrastructure nodes avoids allocating OpenShift Container Platform subscription quota for the nodes that are running those management components.

After adding infrastructure nodes to your OpenShift Container Platform cluster, follow the Installing from the OpenShift Container Platform CLI instructions and add the following configurations to the Operator Lifecycle Manager Subscription and **MultiClusterEngine** custom resource.

1.2.1.5.1. Add infrastructure nodes to the OpenShift Container Platform cluster

Follow the procedures that are described in Creating infrastructure machine sets in the OpenShift Container Platform documentation. Infrastructure nodes are configured with a Kubernetes **taint** and **label** to keep non-management workloads from running on them.

To be compatible with the infrastructure node enablement provided by multicluster engine for Kubernetes operator, ensure your infrastructure nodes have the following **taint** and **label** applied:

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

1.2.1.5.2. Operator Lifecycle Manager Subscription additional configuration

Add the following additional configuration before applying the Operator Lifecycle Manager Subscription:

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

1.2.1.5.3. MultiClusterEngine custom resource additional configuration

Add the following additional configuration before applying the **MultiClusterEngine** custom resource:

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

1.2.2. Install on disconnected networks

You might need to install the multicluster engine for Kubernetes operator on Red Hat OpenShift Container Platform clusters that are not connected to the Internet. The procedure to install on a disconnected engine requires some of the same steps as the connected installation.

Important: You must install multicluster engine for Kubernetes operator on a cluster that does not have Red Hat Advanced Cluster Management for Kubernetes earlier than 2.5 installed. The multicluster engine for Kubernetes operator cannot co-exist with Red Hat Advanced Cluster Management for Kubernetes on versions earlier than 2.5 because they provide some of the same management components. It is recommended that you install multicluster engine for Kubernetes operator on a cluster that has never previously installed Red Hat Advanced Cluster Management. If you are using Red Hat Advanced Cluster Management for Kubernetes at version 2.5.0 or later then multicluster engine for Kubernetes operator is already installed on the cluster with it.

You must download copies of the packages to access them during the installation, rather than accessing them directly from the network during the installation.

- Prerequisites
- Confirm your OpenShift Container Platform installation
- Preparing to install multicluster engine on an infrastructure node

1.2.2.1. Prerequisites

You must meet the following requirements before you install The multicluster engine for Kubernetes operator:

- Red Hat OpenShift Container Platform version 4.8 or later must be deployed in your environment, and you must be logged in with the command line interface (CLI).
- You need access to catalog.redhat.com.
  Note: For managing bare metal clusters, you must have OpenShift Container Platform version 4.8 or later.

  See the OpenShift Container Platform version 4.10, OpenShift Container Platform version 4.8.
- Your Red Hat OpenShift Container Platform CLI must be version 4.8 or later, and configured to run **oc** commands. See Getting started with the CLI for information about installing and configuring the Red Hat OpenShift CLI.
- Your Red Hat OpenShift Container Platform permissions must allow you to create a namespace.
- You must have a workstation with Internet connection to download the dependencies for the operator.

1.2.2.2. Confirm your OpenShift Container Platform installation

- You must have a supported OpenShift Container Platform version, including the registry and storage services, installed and working in your cluster. For information about OpenShift Container Platform version 4.8, see OpenShift Container Platform documentation.

- When and if you are connected, you can ensure that the OpenShift Container Platform cluster is set up correctly by accessing the OpenShift Container Platform web console with the following command:

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

See the following example output:

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

The console URL in this example is: **https:// console-openshift-console.apps.new-coral.purple-chesterfield.com**. Open the URL in your browser and check the result.

If the console URL displays **console-openshift-console.router.default.svc.cluster.local**, set the value for **openshift_master_default_subdomain** when you install OpenShift Container Platform.

1.2.2.3. Installing in a disconnected environment

Important: You need to download the required images to a mirroring registry to install the operators in a disconnected environment. Without the download, you might receive **ImagePullBackOff** errors during your deployment.

Follow these steps to install the multicluster engine for Kubernetes operator in a disconnected environment:

1. Create a mirror registry. If you do not already have a mirror registry, create one by completing the procedure in the Disconnected installation mirroring topic of the Red Hat OpenShift Container Platform documentation.
   If you already have a mirror registry, you can configure and use your existing one.

2. **Note:** For bare metal only, you need to provide the certificate information for the disconnected registry in your **install-config.yaml** file. To access the image in a protected disconnected registry, you must provide the certificate information so the multicluster engine for Kubernetes operator can access the registry.

   a. Copy the certificate information from the registry.

   b. Open the **install-config.yaml** file in an editor.

   c. Find the entry for **additionalTrustBundle: |**.

   d. Add the certificate information after the **additionalTrustBundle** line. The resulting content should look similar to the following example:

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

3. **Important:** Additional mirrors for disconnected image registries are needed if the following Governance policies are required:

   - Container Security Operator policy: The images are located in the source **registry.redhat.io/quay**.
   - Compliance operator policy: The images are located in the source **registry.redhat.io/compliance**
   - **Deprecated** Gatekeeper operator policy: The images are located in the source **registry.redhat.io/rhacm2**
     The Gatekeeper Operator is deprecated to align with Gatekeeper community efforts and releases. Install with a subscription instead.

     See the following example of mirrors lists for all three operators:

```
- 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. Save the **install-config.yaml** file.

5. Create a YAML file that contains the **ImageContentSourcePolicy** with the name **mce-policy.yaml**. **Note:** If you modify this on a running cluster, it causes a rolling restart of all nodes.

```
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. Apply the ImageContentSourcePolicy file by entering the following command:

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

7. Enable the disconnected Operator Lifecycle Manager Red Hat Operators and Community Operators.
   the multicluster engine for Kubernetes operator is included in the Operator Lifecycle Manager Red Hat Operator catalog.

8. Configure the disconnected Operator Lifecycle Manager for the Red Hat Operator catalog. Follow the steps in the Using Operator Lifecycle Manager on restricted networks topic of theRed Hat OpenShift Container Platform documentation.

9. Now that you have the image in the disconnected Operator Lifecycle Manager, continue to install the multicluster engine for Kubernetes operator for Kubernetes from the Operator Lifecycle Manager catalog.

See Installing while connected online for the required steps.

1.2.3. Advanced configuration

The multicluster engine for Kubernetes operator is installed using an operator that deploys all of the required components. The multicluster engine for Kubernetes operator can be further configured during or after installation by adding one or more of the following attributes to the **MultiClusterEngine** custom resource:

1.2.3.1. Local-cluster enablement

A *managed* hub cluster is named a **local-cluster**. If you want the hub cluster to manage itself, you need to change the setting for **spec. disableHubSelfManagement** to **true** to import the existing cluster as a **local-cluster**.

If the setting is not included in the YAML file that defines the custom resource, you need to add it. The hub cluster can only be managed with this option.

1. Create a YAML file named **import-hub.yaml** that is similar to the following example of the default template to use. Replace **namespace** with the name of your project:

```
apiVersion: cluster.open-cluster-management.io/v1
kind: ManagedCluster
metadata:
  labels:
    local-cluster: "true"
    cloud: auto-detect
    vendor: auto-detect
  name: local-cluster
spec:
  hubAcceptsClient: true
  leaseDurationSeconds: 60
```

2. Run the following command to apply the file:

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

A hub cluster that is managed by itself is designated as the **local-cluster** in the list of clusters.

1.2.3.2. Custom image pull secret

If you plan to import Kubernetes clusters that were not created by OpenShift Container Platform or the multicluster engine for Kubernetes operator, generate a secret that contains your OpenShift Container Platform pull secret information to access the entitled content from the distribution registry.

The secret requirements for OpenShift Container Platform clusters are automatically resolved by OpenShift Container Platform and multicluster engine for Kubernetes, so you do not have to create the secret if you are not importing other types of Kubernetes clusters to be managed.

Important: These secrets are namespace-specific, so make sure that you are in the namespace that you use for your engine.

1. Download your OpenShift Container Platform pull secret file from cloud.redhat.com/openshift/install/pull-secret by selecting **Download pull secret**. Your OpenShift Container Platform pull secret is associated with your Red Hat Customer Portal ID, and is the same across all Kubernetes providers.

2. Run the following command to create your secret:

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

   - Replace **secret** with the name of the secret that you want to create.
   - Replace **namespace** with your project namespace, as the secrets are namespace-specific.
   - Replace **path-to-pull-secret** with the path to your OpenShift Container Platform pull secret that you downloaded.

The following example displays the **spec.imagePullSecret** template to use if you want to use a custom pull secret. Replace **secret** with the name of your pull secret:

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

1.2.3.3. Target namespace

The operands can be installed in a designated namespace by specifying a location in the **MultiClusterEngine** custom resource. This namespace is created upon application of the **MultiClusterEngine** custom resource.

Important: If no target namespace is specified, the operator will install to the **multicluster-engine** namespace and will set it in the **MultiClusterEngine** custom resource specification.

The following example displays the **spec.targetNamespace** template that you can use to specify a target namespace. Replace **target** with the name of your destination namespace. **Note:** The **target** namespace cannot be the **default** namespace:

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

1.2.3.4. availabilityConfig

The hub cluster has two availabilities: **High** and **Basic**. By default, the hub cluster has an availability of **High**, which gives hub cluster components a **replicaCount** of **2**. This provides better support in cases of failover but consumes more resources than the **Basic** availability, which gives components a **replicaCount** of **1**.

The following examples shows the **spec.availabilityConfig** template with **Basic** availability:

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

1.2.3.5. nodeSelector

You can define a set of node selectors in the **MultiClusterEngine** to install to specific nodes on your cluster. The following example shows **spec.nodeSelector** to assign pods to nodes with the label **node-role.kubernetes.io/infra**:

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

1.2.3.6. tolerations

You can define a list of tolerations to allow the **MultiClusterEngine** to tolerate specific taints defined on the cluster. The following example shows a **spec.tolerations** that matches a **node-role.kubernetes.io/infra** taint:

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

The previous infra-node toleration is set on pods by default without specifying any tolerations in the configuration. Customizing tolerations in the configuration will replace this default behavior.

1.2.3.7. ManagedServiceAccount add-on (Technology Preview)

By default, the **Managed-ServiceAccount** add-on is disabled. This component when enabled allows you to create or delete a service account on a managed cluster. To install with this add-on enabled, include the following in the **MultiClusterEngine** specification in **spec.overrides**:

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

The **Managed-ServiceAccount** add-on can be enabled after creating **MultiClusterEngine** by editing the resource on the command line and setting the **managedserviceaccount-preview** component to **enabled: true**. Alternatively, you can run the following command and replace <multiclusterengine-name>

with the name of your **MultiClusterEngine** resource.

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

1.2.3.8. Hypershift add-on (Technology Preview)

By default, the **Hypershift** add-on is disabled. To install with this add-on enabled, include the following in the **MultiClusterEngine** values in **spec.overrides**:

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

The **Hypershift** add-on can be enabled after creating **MultiClusterEngine** by editing the resource on the command line, setting the **hypershift-preview** component to **enabled: true**. Alternatively, you can run the following command and replace <multiclusterengine-name> with the name of your **MultiClusterEngine** resource:

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

1.2.4. Uninstalling

When you uninstall multicluster engine for Kubernetes, you see two different levels of the process: A *custom resource removal* and a *complete operator uninstall*. It might take up to five minutes to complete the uninstall process.

- The custom resource removal is the most basic type of uninstall that removes the custom resource of the **MultiClusterEngine** instance but leaves other required operator resources. This level of uninstall is helpful if you plan to reinstall using the same settings and components.
- The second level is a more complete uninstall that removes most operator components, excluding components such as custom resource definitions. When you continue with this step, it removes all of the components and subscriptions that were not removed with the custom resource removal. After this uninstall, you must reinstall the operator before reinstalling the custom resource.

1.2.4.1. Prerequisite: Detach enabled services

Before you uninstall the multicluster engine for Kubernetes engine, you must detach all of the clusters that are managed by that engine. To avoid errors, detach all clusters that are still managed by the engine, then try to uninstall again.

- If you have managed clusters attached, you might see the following message.

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

For more information about detaching clusters, see the *Removing a cluster from management* section by selecting the information for your provider in Creating a cluster.

1.2.4.2. Removing resources by using commands

1. If you have not already. ensure that your OpenShift Container Platform CLI is configured to run **oc** commands. See Getting started with the OpenShift CLI in the OpenShift Container Platform documentation for more information about how to configure the **oc** commands.

2. Change to your project namespace by entering the following command. Replace *namespace* with the name of your project namespace:

```
oc project <namespace>
```

3. Enter the following command to remove the **MultiClusterEngine** custom resource:

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

You can view the progress by entering the following command:

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

4. Enter the following commands to delete the multicluster-engine **ClusterServiceVersion** in the namespace it is installed in:

```
❯ 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
```

The CSV version shown here may be different.

1.2.4.3. Deleting the components by using the console

When you use the RedHat OpenShift Container Platform console to uninstall, you remove the operator. Complete the following steps to uninstall by using the console:

1. In the OpenShift Container Platform console navigation, select **Operators** > **Installed Operators** > **multicluster engine for Kubernetes**.

2. Remove the **MultiClusterEngine** custom resource.

   a. Select the tab for *Multiclusterengine*.

   b. Select the *Options* menu for the MultiClusterEngine custom resource.

   c. Select **Delete MultiClusterEngine**.

3. Run the clean-up script according to the procedure in the following section.
   Tip: If you plan to reinstall the same multicluster engine for Kubernetes version, you can skip the rest of the steps in this procedure and reinstall the custom resource.

4. Navigate to **Installed Operators**.

5. Remove the _ multicluster engine for Kubernetes_ operator by selecting the *Options* menu and selecting **Uninstall operator**.

1.2.4.4. Troubleshooting Uninstall

If the multicluster engine custom resource is not being removed, remove any potential remaining artifacts by running the clean-up script.

a. Copy the following script into a file:

```
#!/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
```

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.11/html/installing/disconnected-installation-mirroring

1.3. MANAGING CREDENTIALS

You can create and manage your cluster credentials. A *credential* is required to create a Red Hat OpenShift Container Platform cluster on a cloud service provider with multicluster engine for Kubernetes operator. The credential stores the access information for a cloud provider. Each provider account requires its own credential, as does each domain on a single provider.

Credentials are stored as Kubernetes secrets. Secrets are copied to the namespace of a managed cluster so that the controllers for the managed cluster can access the secrets. When a credential is updated, the copies of the secret are automatically updated in the managed cluster namespaces.

Note: Changes to the pull secret or SSH keys of cloud provider credentials are not reflected for existing managed clusters, as they have already been provisioned using the original credentials.

Required access: Edit

- Creating a credential for Amazon Web Services
- Creating a credential for Microsoft Azure
- Creating a credential for Google Cloud Platform
- Creating a credential for VMware vSphere
- Creating a credential for Red Hat OpenStack Platform
- Creating a credential for Red Hat Virtualization
- Creating a credential for Red Hat OpenShift Cluster Manager
- Creating a credential for Ansible Automation Platform
- Creating a credential for an on-premises environment

1.3.1. Creating a credential for Amazon Web Services

You need a credential to use multicluster engine for Kubernetes operator console to deploy and manage an Red Hat OpenShift Container Platform cluster on Amazon Web Services (AWS).

Required access: Edit

Note: This procedure must be done before you can create a cluster with multicluster engine for Kubernetes operator.

1.3.1.1. Prerequisites

You must have the following prerequisites before creating a credential:

- A deployed multicluster engine for Kubernetes operator hub cluster
- Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on Amazon Web Services (AWS)
- AWS login credentials, which include access key ID and secret access key. See Understanding and getting your security credentials.
- Account permissions that allow installing clusters on AWS. See Configuring an AWS account for instructions on how to configure.

1.3.1.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

You can optionally add a *Base DNS domain* for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential. See the following steps:

1. Add your *AWS access key ID* for your AWS account. Log in to AWS to find your ID.
2. Provide the contents for your new *AWS Secret Access Key*.
3. If you want to enable a proxy, enter the proxy information:
   - HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
   - HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
   - No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.
   - Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.
4. Enter your *Red Hat OpenShift pull secret*. You can download your pull secret from Pull secret.
5. Add your *SSH private key* and *SSH public key*, which allows you to connect to the cluster. You can use an existing key pair, or create a new one with key generation program.

See Generating an SSH private key and adding it to the agent for more information about how to generate a key.

You can create a cluster that uses this credential by completing the steps in Creating a cluster on Amazon Web Services.

You can edit your credential in the console. If the cluster was created by using this provider connection, then the **<cluster-name>-aws-creds>** secret from **<cluster-namespace>** will get updated with the new credentials.

Note: Updating credentials does not work for cluster pool claimed clusters.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.1.3. Creating an opaque secret by using the API

To create an opaque secret for Amazon Web Services by using the API, apply YAML content in the YAML preview window that is similar to the following example:

```
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)
```

Note: The opaque secret is created in the managed cluster namespace you chose. Hive uses the opaque secret to provision the cluster. When provisioning the cluster by using the Red Hat Advanced Cluster Management console, the credentials you previoulsy created are copied to the managed cluster namespace as the opaque secret.

1.3.2. Creating a credential for Microsoft Azure

You need a credential to use multicluster engine for Kubernetes operator console to create and manage a Red Hat OpenShift Container Platform cluster on Microsoft Azure or on Microsoft Azure Government.

Required access: Edit

Note: This procedure is a prerequisite for creating a cluster with multicluster engine for Kubernetes operator.

1.3.2.1. Prerequisites

You must have the following prerequisites before creating a credential:

- A deployed multicluster engine for Kubernetes operator hub cluster.
- Internet access for your multicluster engine for Kubernetes operator hub cluster so that it can create the Kubernetes cluster on Azure.

- Azure login credentials, which include your Base Domain Resource Group and Azure Service Principal JSON. See azure.microsoft.com.
- Account permissions that allow installing clusters on Azure. See How to configure Cloud Services and Configuring an Azure account for more information.

1.3.2.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console. Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

1. **Optional:** Add a *Base DNS domain* for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential.
2. Select whether the environment for your cluster is **AzurePublicCloud** or **AzureUSGovernmentCloud**. The settings are different for the the Azure Government environment, so ensure that this is set correctly.
3. Add your *Base domain resource group name* for your Azure account. This entry is the resource name that you created with your Azure account. You can find your Base Domain Resource Group Name by selecting **Home** > **DNS Zones** in the Azure interface. See Create an Azure service principal with the Azure CLI to find your base domain resource group name.
4. Provide the contents for your *Client ID*. This value is generated as the **appId** property when you create a service principal with the following command:

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

   Replace *service_principal* with the name of your service principal.
5. Add your *Client Secret*. This value is generated as the **password** property when you create a service principal with the following command:

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

   Replace *service_principal* with the name of your service principal. See Create an Azure service principal with the Azure CLI for more details.
6. Add your *Subscription ID*. This value is the **id** property in the output of the following command:

   ```
   az account show
   ```
7. Add your *Tenant ID*. This value is the **tenantId** property in the output of the following command:

   ```
   az account show
   ```
8. If you want to enable a proxy, enter the proxy information:
   - HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.

- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

9. Enter your *Red Hat OpenShift pull secret*. You can download your pull secret from Pull secret.
10. Add your *SSH private key* and *SSH public key* to use to connect to the cluster. You can use an existing key pair, or create a new pair using a key generation program. See Generating an SSH private key and adding it to the agent for more information about how to generate a key.

You can create a cluster that uses this credential by completing the steps in Creating a cluster on Microsoft Azure.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.2.3. Creating an opaque secret by using the API

To create an opaque secret for Microsoft Azure by using the API instead of the console, apply YAML content in the YAML preview window that is similar to the following example:

```
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}")
```

Note: The opaque secret is created in the managed cluster namespace you chose. Hive uses the opaque secret to provision the cluster. When provisioning the cluster by using the Red Hat Advanced Cluster Management console, the credentials you previoulsy created are copied to the managed cluster namespace as the opaque secret.

1.3.3. Creating a credential for Google Cloud Platform

You need a credential to use multicluster engine for Kubernetes operator console to create and manage a Red Hat OpenShift Container Platform cluster on Google Cloud Platform (GCP).

Required access: Edit

Note: This procedure is a prerequisite for creating a cluster with multicluster engine for Kubernetes operator.

1.3.3.1. Prerequisites

You must have the following prerequisites before creating a credential:

- A deployed multicluster engine for Kubernetes operator hub cluster
- Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on GCP
- GCP login credentials, which include user Google Cloud Platform Project ID and Google Cloud Platform service account JSON key. See Creating and managing projects.
- Account permissions that allow installing clusters on GCP. See Configuring a GCP project for instructions on how to configure an account.

1.3.3.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, for both convenience and security.

You can optionally add a *Base DNS domain* for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential. See the following steps:

1. Add your *Google Cloud Platform project ID* for your GCP account. Log in to GCP to retrieve your settings.
2. Add your *Google Cloud Platform service account JSON key* . See the (https://cloud.google.com/iam/docs/creating-managing-service-accounts) to create your service account JSON key. Follow the steps for the GCP console.
3. Provide the contents for your new *Google Cloud Platform service account JSON key* .
4. If you want to enable a proxy, enter the proxy information:
   - HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
   - HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
   - No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.
   - Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.
5. Enter your *Red Hat OpenShift pull secret* . You can download your pull secret from Pull secret.
6. Add your *SSH private key* and *SSH public key* so you can access the cluster. You can use an existing key pair, or create a new pair using a key generation program.

See Generating an SSH private key and adding it to the agent for more information about how to generate a key.

You can use this connection when you create a cluster by completing the steps in Creating a cluster on Google Cloud Platform.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.3.3. Creating an opaque secret by using the API

To create an opaque secret for Google Cloud Platform by using the API instead of the console, apply YAML content in the YAML preview window that is similar to the following example:

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

Note: The opaque secret is created in the managed cluster namespace you chose. Hive uses the opaque secret to provision the cluster. When provisioning the cluster by using the Red Hat Advanced Cluster Management console, the credentials you previoulsy created are copied to the managed cluster namespace as the opaque secret.

1.3.4. Creating a credential for VMware vSphere

You need a credential to use multicluster engine for Kubernetes operator console to deploy and manage a Red Hat OpenShift Container Platform cluster on VMware vSphere. Only OpenShift Container Platform versions 4.5.x, and later, are supported.

Required access: Edit

Note: This procedure must be done before you can create a cluster with multicluster engine for Kubernetes operator.

1.3.4.1. Prerequisites

You must have the following prerequisites before you create a credential:

- A deployed hub cluster on OpenShift Container Platform version 4.6 or later.
- Internet access for your hub cluster so it can create the Kubernetes cluster on VMware vSphere.
- VMware vSphere login credentials and vCenter requirements configured for OpenShift Container Platform when using installer-provisioned infrastructure. See Installing a cluster on vSphere with customizations. These credentials include the following information:
  - vCenter account privileges.
  - Cluster resources.
  - DHCP available.

- ESXi hosts have time synchronized (for example, NTP).

1.3.4.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

You can optionally add a *Base DNS domain* for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential. See the following steps:

1. Add your *VMware vCenter server fully-qualified host name or IP address* . The value must be defined in the vCenter server root CA certificate. If possible, use the fully-qualified host name.
2. Add your *VMware vCenter username*.
3. Add your *VMware vCenter password*.
4. Add your *VMware vCenter root CA certificate*.
   a. You can download your certificate in the **download.zip** package with the certificate from your VMware vCenter server at: **https://<vCenter_address>/certs/download.zip**. Replace *vCenter_address* with the address to your vCenter server.
   b. Unpackage the **download.zip**.
   c. Use the certificates from the **certs/<platform>** directory that have a **.0** extension. **Tip:** You can use the **ls certs/<platform>** command to list all of the available certificates for your platform.
   Replace ***<platform>*** with the abbreviation for your platform: **lin**, **mac**, or **win**.

   For example: **certs/lin/3a343545.0**

   Best practice: Link together multiple certificates with a **.0** extension using the following command:

   ```
   cat certs/lin/*.0 > ca.crt
   ```

5. Add your *VMware vSphere cluster name* .
6. Add your *VMware vSphere datacenter*.
7. Add your *VMware vSphere default datastore*.
8. For disconnected installations only: Complete the fields in the **Configuration for disconnected installation** subsection with the required information:
   - *Image content source*: This value contains the disconnected registry path. The path contains the hostname, port, and repository path to all of the installation images for disconnected installations. Example: **repository.com:5000/openshift/ocp-release**.
   The path creates an image content source policy mapping in the **install-config.yaml** to the Red Hat OpenShift Container Platform release images. As an example, **repository.com:5000** produces this **imageContentSource** content:

```
imageContentSources:
- 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*: This value provides the contents of the certificate file that is required to access the mirror registry.
  Note: If you are deploying managed clusters from a hub that is in a disconnected environment, and want them to be automatically imported post install, add an Image Content Source Policy to the **install-config.yaml** file by using the **YAML** editor. A sample entry is shown in the following example:

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

9. If you want to enable a proxy, enter the proxy information:
   - HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
   - HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
   - No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.
   - Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.
10. Enter your *Red Hat OpenShift pull secret*. You can download your pull secret from Pull secret.
11. Add your *SSH private key* and *SSH public key*, which allows you to connect to the cluster. You can use an existing key pair, or create a new one with key generation program. See Generating a key pair for cluster node SSH access for more information.

You can create a cluster that uses this credential by completing the steps in Creating a cluster on VMware vSphere.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.4.3. Creating an opaque secret by using the API

To create an opaque secret for VMware vSphere by using the API instead of the console, apply YAML content in the YAML preview window that is similar to the following example:

```
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}")
```

Note: The opaque secret is created in the managed cluster namespace you chose. Hive uses the opaque secret to provision the cluster. When provisioning the cluster by using the Red Hat Advanced Cluster Management console, the credentials you previoulsy created are copied to the managed cluster namespace as the opaque secret.

1.3.5. Creating a credential for Red Hat OpenStack

You need a credential to use multicluster engine for Kubernetes operator console to deploy and manage a Red Hat OpenShift Container Platform cluster on Red Hat OpenStack Platform. Only OpenShift Container Platform versions 4.5.x, and later, are supported.

Notes: This procedure must be done before you can create a cluster with multicluster engine for Kubernetes operator.

1.3.5.1. Prerequisites

You must have the following prerequisites before you create a credential:

- A deployed hub cluster on OpenShift Container Platform version 4.6 or later.
- Internet access for your hub cluster so it can create the Kubernetes cluster on Red Hat OpenStack Platform.
- Red Hat OpenStack Platform login credentials and Red Hat OpenStack Platform requirements configured for OpenShift Container Platform when using installer-provisioned infrastructure. See Installing a cluster on OpenStack with customizations .
- Download or create a **clouds.yaml** file for accessing the CloudStack API. Within the **clouds.yaml** file:
  - Determine the cloud auth section name to use.
  - Add a line for the **password**, immediately following the **username** line.

1.3.5.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, for both convenience and added security.

1. Add your Red Hat OpenStack Platform **clouds.yaml** file contents. The contents of the **clouds.yaml** file, including the password, provide the required information for connecting to the

Red Hat OpenStack Platform server. The file contents must include the password, which you add to a new line immediately after the **username**.

2. For configurations that use an internal certificate authority, modify your **clouds.yaml** file to reference the final location for the certificate bundle within the Hive deployer pod. Hive mounts the certificate bundle secret to **/etc/openstack-ca** within the deployer pod. The files inside of that directory correspond to the keys in the secret that is provided when you create the cluster. Assuming that the key **ca.crt** is used in the secret, add the **cacert** parameter to the **clouds.yaml** file as shown in the following example:

```
clouds:
  openstack:
    auth:
      auth_url: https://openstack.example.local:13000
      username: "svc-openshift"
      project_id: aa0owet0wfwerj
      user_domain_name: "idm"
      password: REDACTED
      region_name: "regionOne"
      interface: "public"
      identity_api_version: 3
      cacert: /etc/openstack-ca/ca.crt
```

3. Add your Red Hat OpenStack Platform cloud name. This entry is the name specified in the cloud section of the **clouds.yaml** to use for establishing communication to the Red Hat OpenStack Platform server.

4. You can optionally add a Base DNS domain for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential.

5. For disconnected installations only: Complete the fields in the **Configuration for disconnected installation** subsection with the required information:

   - *Cluster OS image*: This value contains the URL to the image to use for Red Hat OpenShift Container Platform cluster machines.

   - *Image content sources*: This value contains the disconnected registry path. The path contains the hostname, port, and repository path to all of the installation images for disconnected installations. Example: **repository.com:5000/openshift/ocp-release**. The path creates an image content source policy mapping in the **install-config.yaml** to the Red Hat OpenShift Container Platform release images. As an example, **repository.com:5000** produces this **imageContentSource** content:

```
imageContentSources:
- 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*: This value provides the contents of the certificate file that is required to access the mirror registry.
  Note: If you are deploying managed clusters from a hub that is in a disconnected environment, and want them to be automatically imported post install, add an Image Content Source Policy to the **install-config.yaml** file by using the **YAML** editor. A sample entry is shown in the following example:

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

6. If you want to enable a proxy, enter the proxy information:
   - HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
   - HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
   - No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk ***** to bypass the proxy for all destinations.
   - Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.
7. Enter your Red Hat OpenShift Pull Secret. You can download your pull secret from Pull secret.
8. Add your SSH Private Key and SSH Public Key, which allows you to connect to the cluster. You can use an existing key pair, or create a new one with key generation program. See Generating a key pair for cluster node SSH access for more information.
9. Click **Create**.
10. Review the new credential information, then click **Add**. When you add the credential, it is added to the list of credentials.

You can create a cluster that uses this credential by completing the steps in Creating a cluster on Red Hat OpenStack Platform.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.5.3. Creating an opaque secret by using the API

To create an opaque secret for Red Hat OpenStack Platform by using the API instead of the console, apply YAML content in the YAML preview window that is similar to the following example:

```
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)
```

Note: The opaque secret is created in the managed cluster namespace you chose. Hive uses the opaque secret to provision the cluster. When provisioning the cluster by using the Red Hat Advanced Cluster Management console, the credentials you previoulsy created are copied to the managed cluster namespace as the opaque secret.

1.3.6. Creating a credential for Red Hat Virtualization

You need a credential to use multicluster engine for Kubernetes operator console to deploy and manage a Red Hat OpenShift Container Platform cluster on Red Hat Virtualization.

Note: This procedure must be done before you can create a cluster with multicluster engine for Kubernetes operator.

1.3.6.1. Prerequisites

You must have the following prerequisites before you create a credential:

- A deployed hub cluster on OpenShift Container Platform version 4.7 or later.
- Internet access for your hub cluster so it can create the Kubernetes cluster on Red Hat Virtualization.
- Red Hat Virtualization login credentials for a configured Red Hat Virtualization environment. See Installation Guide in the Red Hat Virtualization documentation. The following list shows the required information:
  - oVirt URL
  - oVirt fully-qualified domain name (FQDN)
  - oVirt username
  - oVirt password
  - oVirt CA/Certificate
- Optional: Proxy information, if you are enabling a proxy.
- Red Hat OpenShift Container Platform pull secret information. You can download your pull secret from Pull secret.
- SSH private and public keys for transferring information for the final cluster.
- Account permissions that allow installing clusters on oVirt.

1.3.6.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, for both convenience and added security.

1. Add the basic information for your new credential. You can optionally add a Base DNS domain, which is automatically populated in the correct field when you create a cluster with this credential. If you do not add it to the credential, you can add it when you create the cluster.

2. Add the required information for your Red Hat Virtualization environment.

3. If you want to enable a proxy, enter the proxy information:

   - HTTP Proxy URL: The URL that should be used as a proxy for **HTTP** traffic.

   - HTTPS Proxy URL: The secure proxy URL that should be used when using **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.

   - No Proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk * to bypass the proxy for all destinations.

4. Enter your Red Hat OpenShift Container Platform pull secret. You can download your pull secret from Pull secret.

5. Add your SSH Private Key and SSH Public Key, which allows you to connect to the cluster. You can use an existing key pair, or create a new one with a key generation program. See Generating akey pair for cluster node SSH access for more information.

6. Review the new credential information, then click **Add**. When you add the credential, it is added to the list of credentials.

You can create a cluster that uses this credential by completing the steps in Creating a cluster on Red Hat Virtualization.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.7. Creating a credential for Red Hat OpenShift Cluster Manager

Add an OpenShift Cluster Manager credential so that you can discover clusters.

Required access: Administrator

1.3.7.1. Prerequisites

You need access to a console.redhat.com account. Later you will need the value that can be obtained from console.redhat.com/openshift/token.

1.3.7.2. Managing a credential by using the console

You need to add your credential to discover clusters. To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

Your OpenShift Cluster Manager API token can be obtained from console.redhat.com/openshift/token.

You can edit your credential in the console.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

If your credential is removed, or your OpenShift Cluster Manager API token expires or is revoked, then the associated discovered clusters are removed.

1.3.8. Creating a credential for Ansible Automation Platform

You need a credential to use multicluster engine for Kubernetes operator console to deploy and manage an Red Hat OpenShift Container Platform cluster that is using Red Hat Ansible Automation Platform.

Required access: Edit

Note: This procedure must be done before you can create an Ansible job template to enable automation on a cluster.

1.3.8.1. Prerequisites

You must have the following prerequisites before creating a credential:

- A deployed multicluster engine for Kubernetes operator hub cluster
- Internet access for your multicluster engine for Kubernetes operator hub cluster
- Ansible login credentials, which includes Ansible Tower hostname and OAuth token; see Credentials for Ansible Tower.
- Account permissions that allow you to install hub clusters and work with Ansible. Learn more about Ansible users.

1.3.8.2. Managing a credential by using the console

To create a credential from the multicluster engine for Kubernetes operator console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

The Ansible Token and host URL that you provide when you create your Ansible credential are automatically updated for the automations that use that credential when you edit the credential. The updates are copied to any automations that use that Ansible credential, including those related to cluster lifecycle, governance, and application management automations. This ensures that the automations continue to run after the credential is updated.

You can edit your credential in the console. Ansible credentials are automatically updated in your automation that use that credential when you update them in the credential.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.3.9. Creating a credential for an on-premises environment

You need a credential to use the console to deploy and manage a Red Hat OpenShift Container Platform cluster in an on-premises environment. The credential specifies the connections that are used for the cluster.

Required access: Edit

- Prerequisites
- Managing a credential by using the console

1.3.9.1. Prerequisites

You need the following prerequisites before creating a credential:

- A hub cluster that is deployed.
- Internet access for your hub cluster so it can create the Kubernetes cluster on your infrastructure environment.
- For a disconnected environment, you must have a configured mirror registry where you can copy the release images for your cluster creation. See Mirroring images for a disconnected installation in the OpenShift Container Platform documentation for more information.
- Account permissions that support installing clusters on the on-premises environment.

1.3.9.2. Managing a credential by using the console

To create a credential from the console, complete the steps in the console.

Start at the navigation menu. Click **Credentials** to choose from existing credential options. **Tip:** Create a namespace specifically to host your credentials, both for convenience and added security.

1. Select **Host inventory** for your credential type.
2. You can optionally add a *Base DNS domain* for your credential. If you add the base DNS domain to the credential, it is automatically populated in the correct field when you create a cluster with this credential. If you do not add the DNS domain, you can add it when you create your cluster.
3. Enter your *Red Hat OpenShift pull secret*. You can download your pull secret from Pull secret. See Using image pull secrets for more information about pull secrets.
4. Select **Add** to create your credential.

When you are no longer managing a cluster that is using a credential, delete the credential to protect the information in the credential. Select **Actions** to delete in bulk, or select the options menu beside the credential that you want to delete.

1.4. THE MULTICLUSTER ENGINE FOR KUBERNETES OPERATOR CLUSTER LIFECYCLE OVERVIEW

The multicluster engine for Kubernetes operator is a software operator that enhances cluster fleet management. The multicluster engine for Kubernetes operator supports Red Hat OpenShift Container Platform and Kubernetes cluster lifecycle management across clouds and data centers.

See the following documentation:

- Cluster lifecycle architecture
- Managing credentials overview
- Release images
  - Maintaining a custom list of release images while disconnected
  - Creating an infrastructure environment
- Creating a cluster
  - Creating a cluster with the CLI
  - Configuring additional manifests during cluster creation
  - Creating a cluster on Amazon Web Services
  - Creating a cluster on Microsoft Azure
  - Creating a cluster on Google Cloud Platform
  - Creating a cluster on VMware vSphere
  - Creating a cluster on Red Hat OpenStack Platform
  - Creating a cluster on Red Hat Virtualization
  - Creating a cluster in an on-premise environment
  - Creating a cluster in a proxy environment
- Importing a target managed cluster to the hub cluster
  - Importing an existing cluster with the console
  - Importing a managed cluster with the CLI
  - Importing a cluster with a custom ManagedClusterImageRegistry CRD
- Hibernating a created cluster (Technology Preview)
- Scaling managed clusters
  - Scaling with MachinePool (Technology Preview)
  - Scaling hosts to an infrastructure environment
- Enabling cluster proxy add-ons
- Configuring a specific cluster management role
- Managing cluster labels
- Configuring Ansible Tower tasks to run on managed clusters
- Creating and managing ManagedClusterSets

    - Creating a ManagedClusterSet
    - Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet
    - Creating a ManagedClusterSetBinding resource
    - Adding a cluster to a ManagedClusterSet
    - Using ManagedClusterSets with Placement
    - Using taints and tolerations to place managed clusters
    - Removing a cluster from a ManagedClusterSet
- Managing cluster pools (Technology Preview)
    - Creating a cluster pool
    - Claiming clusters from cluster pools
    - Updating the cluster pool release image
    - Scaling cluster pools
    - Destroying a cluster pool
- ClusterClaims
    - List existing ClusterClaims
    - Create custom ClusterClaims
- Enabling ManagedServiceAccount
- Upgrading your cluster
- Removing a cluster from management

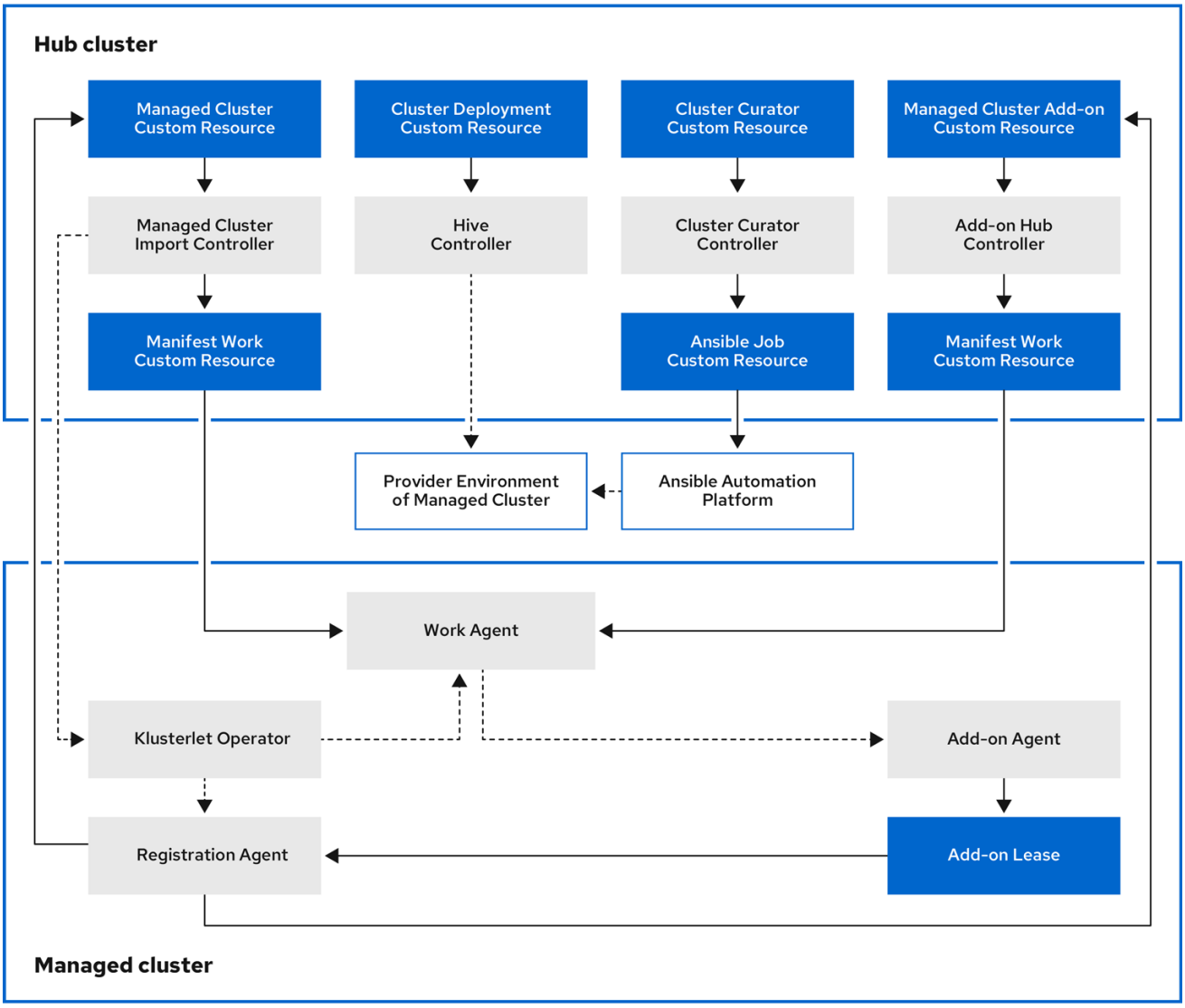
1.4.1. Cluster lifecycle architecture

The multicluster engine for Kubernetes uses two main types of clusters: *hub clusters* and *managed clusters*.

The hub cluster is the main cluster with the multicluster engine installed on it. You can create, manage, and monitor other Kubernetes clusters with the hub cluster. The managed clusters are Kubernetes clusters that are managed by the hub cluster. You can create some clusters by using the hub cluster, while you can also import existing clusters to be managed by the hub cluster.

When you create a managed cluster by using the multicluster engine for Kubernetes operator, the cluster is created using the Red Hat OpenShift Container Platform cluster installer with the Hive resource. You can find more information about the process of installing clusters with the OpenShift Container Platform installer by reading OpenShift Container Platform installation overview in the OpenShift Container Platform documentation.

The following diagram shows the components that are installed with the multicluster engine for Kubernetes operator for cluster management:



The components of the cluster lifecycle management architecture include the following items:

1.4.1.1. Hub cluster

- The *managed cluster import controller* deploys the klusterlet operator to the managed clusters.
- The *Hive controller* provisions the clusters that you create by using the multicluster engine for Kubernetes operator. The Hive Controller also destroys managed clusters that were created by the multicluster engine for Kubernetes operator.
- The *cluster curator controller* creates the Ansible jobs as the pre-hook or post-hook to configure the cluster infrastructure environment when creating or upgrading managed clusters.
- When a managed cluster add-on is enabled on the hub cluster, its *add-on hub controller* is deployed on the hub cluster. The *add-on hub controller* deploys the *add-on agent* to the managed clusters.

1.4.1.2. Managed cluster

- The *klusterlet operator* deploys the registration and work controllers on the managed cluster.

- The *Registration Agent* registers the managed cluster and the managed cluster add-ons with the hub cluster. The Registration Agent also maintains the status of the managed cluster and the managed cluster add-ons. The following permissions are automatically created within the Clusterrole to allow the managed cluster to access the hub cluster:
  - Allows the agent to get or update its owned cluster that the hub cluster manages
  - Allows the agent to update the status of its owned cluster that the hub cluster manages
  - Allows the agent to rotate its certificate
  - Allows the agent to **get** or **update** the **coordination.k8s.io** lease
  - Allows the agent to **get** its managed cluster add-ons
  - Allows the agent to update the status of its managed cluster add-ons
- The *work agent* applies the manifest work to the managed cluster. The following permission is automatically created within the Clusterrole to allow the managed cluster to access the hub cluster:
  - Allows the agent to send events to the hub cluster
  - Allows the agent to **get** or **update** the **manifestworks** resource
  - Allows the agent to update the status of **manifestworks** resource
- When a managed cluster add-on is created in a managed cluster namespace on the hub cluster, its hub controller manifests its agent deployment resources with manifest work. The work agent then applies the manifest work to the managed cluster to deploy the add-on agent.

To continue adding and managing clusters, see the multicluster engine for Kubernetes operator cluster lifecycle overview.

1.4.2. Release images

When you create a cluster on a provider by using multicluster engine for Kubernetes operator, you must specify a release image to use for the new cluster. The release image specifies which version of Red Hat OpenShift Container Platform is used to build the cluster.

The files that reference the release images are YAML files that are maintained in the **acm-hive-openshift-releases** GitHub repository. Red Hat Advanced Cluster Management uses those files to create the list of the available release images in the console. This includes the latest fast channel images from OpenShift Container Platform. The console only displays the latest release images for the three latest versions of OpenShift Container Platform. For example, you might see the following release images displayed in the console options:

- quay.io/openshift-release-dev/ocp-release:4.6.23-x86_64
- quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64

Note: Only release images with the label of: **visible: 'true'** are available to select when creating clusters in the console. An example of this label in a **ClusterImageSet** resource is provided in the following content:

```
apiVersion: config.openshift.io/v1
kind: ClusterImageSet
```

```
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.10.1-x86-64-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64
```

Additional release images are stored, but are not visible in the console. To view all of the available release images, run **kubectl get clusterimageset** in your CLI. Only the latest versions are in the console to encourage the creation of clusters with the latest release images. In some cases, you might need to create a cluster that is a specific version, which is why the older versions are available. Red Hat Advanced Cluster Management uses those files to create the list of the available release images in the console. This includes the latest fast channel images from OpenShift Container Platform.

The repository contains the **clusterImageSets** directory and the **subscription** directory, which are the directories that you use when working with the release images.

The **clusterImageSets** directory contains the following directories:

- Fast: Contains files that reference the latest versions of the release images for each OpenShift Container Platform version that is supported. The release images in this folder are tested, verified, and supported.
- Releases: Contains files that reference all of the release images for each OpenShift Container Platform version (stable, fast, and candidate channels) **Note:** These releases have not all been tested and determined to be stable.
- Stable: Contains files that reference the latest two stable versions of the release images for each OpenShift Container Platform version that is supported.

Note: By default, the current list of release images is updated one time an hour. After upgrading the product, it may take up to an hour for the list to reflect the recommended release image versions for the new version of the product.

You can curate your own **ClusterImageSets** in three ways:

The first step for any of the three ways is to disable the included subscription that automatically updates the latest fast channel images. The automatic curation of the latest fast **ClusterImageSets** can be disabled by using an installer parameter on the **multiclusterhub** resource. By toggling the **spec.disableUpdateClusterImageSets** parameter between **true** and **false**, the subscription installed with Red Hat Advanced Cluster Management is disabled or enabled, respectively. If you want to curate your own images, set the **spec.disableUpdateClusterImageSets** to **true** to disable the subscription.

Option 1: Specify the image reference for the specific **ClusterImageSet** that you want to use in the console when creating a cluster. Each new entry you specify persists and is available for all future cluster provisions. An example of an entry is: **quay.io/openshift-release-dev/ocp-release:4.6.8-x86_64**.

Option 2: Manually create and apply a **ClusterImageSets** YAML file from the **acm-hive-openshift-releases** GitHub repository.

Option 3: Follow the **README.md** in the **acm-hive-openshift-releases** GitHub repository to enable automatic updates of **ClusterImageSets** from a forked GitHub repository.

The **subscription** directory contains files that specify where the list of release images is pulled from.

The default release images for Red Hat Advanced Cluster Management are provided in a Quay.io directory.

The images are referenced by the files in the acm-hive-openshift-releases GitHub repository for release 2.5.

1.4.2.1. Creating a release image to deploy a cluster on a different architecture

You can create a cluster on an architecture that is different from the architecture of the hub cluster by manually creating a release image that contains the files for both architectures.

For example, you might need to create an **x86_64** cluster from a hub cluster that is running on the **ppc64le**, **aarch64**, or **s390x** architecture. If you create the release image with both sets of files, the cluster creation succeeds because the new release image enables the OpenShift Container Platform release registry to provide a multi-architecture image manifest.

OpenShift Container Platform 4.11 and later supports multiple architectures by default. You can use the following **clusterImageSet** to provision a cluster:

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: 'true'
  name: img4.12.0-multi-appsub
spec:
  releaseImage: quay.io/openshift-release-dev/ocp-release:4.12.0-multi
```

To create the release image for OpenShift Container Platform images that do not support multiple architectures, complete steps similar to the following example for your architecture type:

1. From the OpenShift Container Platform release registry, create a manifest list that includes **x86_64**, **s390x**, **aarch64**, and **ppc64le** release images.

   a. Pull the manifest lists for both architectures in your environment from the Quay repository using the following example commands:

   ```
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-ppc64le
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-s390x
   podman pull quay.io/openshift-release-dev/ocp-release:4.10.1-aarch64
   ```

   b. Log in to your private repository where you maintain your images:

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

   Replace **private-repo** with the path to your repository.

   c. Add the release image manifest to your private repository by running the following commands that apply to your environment:

   ```
   podman push quay.io/openshift-release-dev/ocp-release:4.10.1-x86_64 <private-
   repo>/ocp-release:4.10.1-x86_64
   podman push quay.io/openshift-release-dev/ocp-release:4.10.1-ppc64le <private-
   ```

```
repo>/ocp-release:4.10.1-ppc64le
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-s390x <private-
repo>/ocp-release:4.10.1-s390x
podman push quay.io/openshift-release-dev/ocp-release:4.10.1-aarch64 <private-
repo>/ocp-release:4.10.1-aarch64
```

Replace **private-repo** with the path to your repository.

d. Create a manifest for the new information:

```
podman manifest create mymanifest
```

e. Add references to both release images to the manifest list:

```
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-x86_64
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-ppc64le
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-s390x
podman manifest add mymanifest <private-repo>/ocp-release:4.10.1-aarch64
```

Replace **private-repo** with the path to your repository.

f. Merge the list in your manifest list with the existing manifest:

```
podman manifest push mymanifest docker://<private-repo>/ocp-release:4.10.1
```

Replace **private-repo** with the path to your repository.

2. On the hub cluster, create a release image that references the manifest in your repository.

a. Create a YAML file that contains information that is similar to the following example:

```
apiVersion: hive.openshift.io/v1
kind: ClusterImageSet
metadata:
  labels:
    channel: fast
    visible: "true"
  name: img4.10.1-appsub
spec:
  releaseImage: <private-repo>/ocp-release:4.10.1
```

Replace **private-repo** with the path to your repository.

b. Run the following command on your hub cluster to apply the changes:

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

Replace **file-name** with the name of the YAML file that you just created.

3. Select the new release image when you create your OpenShift Container Platform cluster.

4. If you deploy the managed cluster using the Red Hat Advanced Cluster Management console, specify the architecture for the managed cluster in the *Architecture* field during the cluster creation process.

The creation process uses the merged release images to create the cluster.

1.4.2.2. Maintaining a custom list of release images while disconnected

In some cases, you need to maintain a custom list of release images when the hub cluster has no Internet connection. You can create your own custom list of release images that are available when creating a cluster. Complete the following steps to manage your available release images while disconnected:

1. While you are on a connected system, navigate to the acm-hive-openshift-releases GitHub repository to access the cluster image sets that are available for version 2.5.

2. Copy the **clusterImageSets** directory to a system that can access the disconnected multicluster engine for Kubernetes operator hub cluster.

3. Add the mapping between the managed cluster and the disconnected repository with your cluster image sets by completing the following steps that fits your managed cluster:

   - For an OpenShift Container Platform managed cluster, see Configuring image registry repository mirroring for information about using your **ImageContentSourcePolicy** object to complete the mapping.

   - For a managed cluster that is not an OpenShift Container Platform cluster, use the **ManageClusterImageRegistry** CRD to override the location of the image sets. See Importing a cluster with a custom ManagedClusterImageRegistry CRD for information about how to override the cluster for the mapping.

4. Add the YAML files for the images that you want available when you create a cluster by using the Red Hat Advanced Cluster Management console by manually adding the **clusterImageSet** YAML content.

5. Modify the **clusterImageSet** YAML files for the remaining OpenShift Container Platform release images to reference the correct offline repository where you store the images. Your updates should resemble the following example:

   ```
   apiVersion: hive.openshift.io/v1
   kind: ClusterImageSet
   metadata:
     name: img4.4.0-rc.6-x86-64
   spec:
     releaseImage: IMAGE_REGISTRY_IPADDRESS_or_DNSNAME/REPO_PATH/ocp-release:4.4.0-rc.6-x86_64
   ```

   Ensure that the images are loaded in the offline image registry that is referenced in the YAML file.

6. Create each of the **clusterImageSets** by entering the following command for each YAML file:

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

   Replace **clusterImageSet_FILE** with the name of the cluster image set file. For example:

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

   After running this command for each resource you want to add, the list of available release images will be available.

7. Alternately you can paste the image URL directly in the create cluster console in Red Hat Advanced Cluster Management. Adding the image URL creates new clusterImageSets if they do not exist.

8. View the list of currently available release images in the Red Hat Advanced Cluster Management console when you are creating a cluster.

1.4.3. Creating an infrastructure environment

You can use the console to create an infrastructure environment to manage your hosts and create clusters on those hosts.

- Prerequisites
- Enable Central Infrastructure Management service
  - Manually create the Provisioning custom resource (CR)
  - Enable Central Infrastructure Management on Amazon Web Services
- Creating your infrastructure environment with the console

Infrastructure environments support the following features:

- Zero-touch provisioning of clusters: Deploy clusters using a script. See Installing GitOps ZTP in a disconnected environment in the Red Hat OpenShift Container Platform documentation for more information.
- Late binding: Add a node to an existing cluster. An infrastructure administrator can boot the host, and the cluster creator can bind the host to the existing cluster at a later time. Administrator privileges are not required for the cluster creator to access the infrastructure when using late binding.
- Dual stack: Deploy clusters that have both IPv4 and IPv6 addresses. Dual stack uses the **OVN-Kubernetes** networking implementation to support multiple subnets.
- Add remote worker nodes: Add remote worker nodes to your clusters after they are created and running, which provides flexibility of adding nodes in other locations for backup purposes.
- Static IP using NMState: Use the NMState API to define static IP addresses for your environment.

1.4.3.1. Prerequisites

See the following prerequisites before creating an infrastructure environment:

- You must have OpenShift Container Platform deployed on your hub cluster.
- You need Internet access for your hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the Internet (disconnected) to retrieve the required images for creating the environment.
- You need a configured instance of the Central Infrastructure Management (CIM) feature on your hub cluster.
- You must have the Bare Metal Operator installed on your hub cluster.

- You need an OpenShift Container Platform pull secret. See Using image pull secrets for more information.
- You need your SSH key that is in your **~/.ssh/id_rsa.pub** file, by default.
- You need a configured storage class.
- **Disconnected environment only:** Complete the procedure for Clusters at the network far edge in the OpenShift Container Platform documentation.

1.4.3.2. Enabling the Central Infrastructure Management service

The Central Infrastructure Management service is provided with multicluster engine for Kubernetes and deploys OpenShift Container Platform clusters. CIM is deployed when you enable the **MultiClusterHub** Operator on the hub cluster, but must be enabled.

To enable the CIM service, complete the following steps:

Important: Only if your hub cluster is installed on one of the following platforms: bare metal, Red Hat OpenStack Platform, VMware vSphere, or was installed by using the user-provisioned infrastructure (UPI) method and the platform is **None**, complete the following step. Skip this step if your hub cluster is on any other platform.

1. Modify the **Provisioning** resource to allow the Bare Metal Operator to watch all namespaces by running the following command:

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

2. **For disconnected environments:** Create a **ConfigMap** in the *same namespace* as your infrastructure operator to specify the values for **ca-bundle.crt** and **registries.conf** for your mirror registry. Your file **ConfigMap** should resemble the following example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: <mirror-config>
  namespace: "<infrastructure-operator-namespace>"
  labels:
    app: assisted-service
data:
  ca-bundle.crt: |
    -----BEGIN CERTIFICATE-----
    certificate contents
    -----END CERTIFICATE-----

  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"
```

-

1.4.3.2.1. Creating the *AgentServiceConfig* custom resource

Create the **AgentServiceConfig** custom resource by completing the following steps:

1. **For disconnected environments only:** Save the following **YAML** content in the **agent_service_config.yaml** file and replace the values as needed:

```
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>
  unauthenticatedRegistries:
    - <unauthenticated_registry>
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size>
  osImages:
    - openshiftVersion: "<ocp_version>"
      version: "<ocp_release_version>"
      url: "<iso_url>"
      cpuArchitecture: "x86_64"
```

   Replace **mirror_config** with the name of the **ConfigMap** that contains your mirror registry configuration details.

   Include the optional **unauthenticated_registry** parameter if you are using a mirror registry that does not require authentication. Entries on this list are not validated or required to have an entry in the pull secret.

2. **For connected environments only:** Save the following **YAML** content in the **agent_service_config.yaml** file:

```
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>
  imageStorage:
    accessModes:
    - ReadWriteOnce
    resources:
      requests:
        storage: <img_volume_size>
```

+ Replace **db_volume_size** with the volume size for the **databaseStorage** field, for example **10Gi**. This value specifies how much storage is allocated for storing files such as database tables and database views for the clusters. You might need to use a higher value if there are many clusters.

+ Replace **fs_volume_size** with the size of the volume for the **filesystemStorage** field, for example **200M** per cluster and **2-3Gi** per supported OpenShift Container Platform version. The minimum value that is required is **100Gi**. This value specifies how much storage is allocated for storing logs, manifests, and **kubeconfig** files for the clusters. You might need to use a higher value if there are many clusters.

+ Replace **img_volume_size** with the size of the volume for the **imageStorage** field, for example **2Gi** per operating system image. The minimum size is **50Gi**. This value specifies how much storage is allocated for the images of the clusters. You need to allow 1 GB of image storage for each instance of Red Hat Enterprise Linux CoreOS that is running. You might need to use a higher value if there are many clusters and instances of Red Hat Enterprise Linux CoreOS.

+ Replace **ocp_version** with the OpenShift Container Platform version to install.

+ Replace **ocp_release_version** with the specific install version, for example, **49.83.202103251640-0**.

+ Replace **iso_url** with the ISO url, for example, **https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/rhcos-4.10.3-x86_64-live.x86_64.iso**. You can find other values at: https://mirror.openshift.com/pub/openshift-v4/x86_64/dependencies/rhcos/4.10/4.10.3/.

1. Create the **AgentServiceConfig** custom resource by running the following command:

   ```
   oc create -f agent_service_config.yaml
   ```

   The output might resemble the following example:

   ```
   agentserviceconfig.agent-install.openshift.io/agent created
   ```

Your CIM service is configured. You can verify that it is healthy by checking the **assisted-service** and **assisted-image-service** deployments and ensuring that their pods are ready and running.

1.4.3.2.2. Manually create the Provisioning custom resource (CR)

Manually create a **Provisioning** CR to enable services for automated provisioning by using the following command:

```
oc create -f provisioning-configuration.yaml
```

Your CR might resemble the following sample:

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

1.4.3.2.3. Enabling Central Infrastructure Management on Amazon Web Services

If you are running your hub cluster on Amazon Web Services and want to enable the CIM service, complete the following additional steps after Enabling CIM:

1. Make sure you are logged in at the hub and find the unique domain configured on the **assisted-image-service** by running the following command:

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

   Your domain might resemble the following example: **assisted-image-service-multicluster-engine.apps.<yourdomain>.com**

2. Make sure you are logged in at the hub and create a new **IngressController** with a unique domain using the **NLB type** parameter. See the following example:

   ```
   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. Add **<yourdomain>** to the **domain** parameter in **IngressController** by replacing **<domain>** in **nlb-apps.<domain>.com** with **<yourdomain>**.

4. Apply the new **IngressController** by using the following command:

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

5. Make sure that the value of the **spec.domain** parameter of the new **IngressController** is not in conflict with an existing IngressController by completing the following steps:
   a. List all **IngressControllers** by running the following command:

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

   b. Run the following command on each of the **IngressControllers**, except the **ingress-controller-with-nlb** that you just created:

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

      If the **spec.domain** report is missing, add a default domain that matches all of the routes that are exposed in the cluster except **nlb-apps.<domain>.com**.

      If the **spec.domain** report is provided, make sure that the **nlb-apps.<domain>.com** route is excluded from the specified range.

6. Run the following command to edit the **assisted-image-service** route to use the **nlb-apps** location:

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

7. Add the following lines to the **assisted-image-service** route:

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

8. In the **assisted-image-service** route, find the URL value of **spec.host**. The URL might resemble the following example:
   assisted-image-service-multicluster-engine.apps.<yourdomain>.com

9. Replace **apps** in the URL with **nlb-apps** to match the domain configured in the new **IngressController**.

To verify that the CIM service is enabled on Amazon Web Services, complete the following steps:

1. Run the following command to verify that the pods are healthy:

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

2. Create a new infrastructure environment and ensure that the download URL uses the new **nlb-apps** URL.

1.4.3.3. Creating your infrastructure environment with the console

To create an infrastructure environment from the console, complete the following steps:

1. From the navigation menu, navigate to **Infrastructure** > **Host inventory** and click **Create infrastructure environment**.

2. Add the following information to your infrastructure environment settings:

   - Name: A unique name for your infrastructure environment.
   - Network type: Specifies which types of hosts can be added to your infrastructure environment. You can only use the static IP option when you are using bare metal hosts.
   - Location: Specifies the geographic location of the host. The geographic location can be used to easily determine where your data on a cluster is stored when you are creating the cluster.
   - Labels: Optional field where you can add labels to the infrastructure environment so you can more easily find and group the environment with other environments that share a characteristic. The selections that you made for the network type and location are automatically added to the list of labels.
   - Pull secret: Your OpenShift Container Platform pull secret that enables you to access the OpenShift Container Platform resources.
   - SSH public key: The SSH key that enables the secure communication with the hosts. This is generally in your ~/**.ssh/id_rsa.pub** file, by default.
   - If you want to enable proxy settings across all of your clusters, select the setting to enable it. This requires that you enter the following information:
     - HTTP Proxy URL: The URL that should be used when accessing the discovery service.
     - HTTPS Proxy URL: The secure proxy URL that should be used when accessing the discovery service. Note that the format must be **http**, as **https** is not yet supported.
     - No Proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add and asterisk * to bypass the proxy for all destinations.

You can now continue by adding hosts to your infrastructure environment.

To access an infrastructure environment, select **Infrastructure** > **Host inventory** in the console. Select your infrastructure environment from the list to view the details and hosts for that infrastructure environment.

1.4.4. Creating a cluster

Learn how to create Red Hat OpenShift Container Platform clusters across cloud providers with multicluster engine for Kubernetes operator.

multicluster engine for Kubernetes operator uses the Hive operator that is provided with OpenShift Container Platform to provision clusters for all providers except the on-premises clusters and hosted control planes. When provisioning the on-premises clusters, multicluster engine for Kubernetes operator uses the Central Infrastructure Management (CIM) and Assisted Installer function that are provided with OpenShift Container Platform. The hosted clusters for hosted control planes are provisioned by using the HyperShift operator.

- Configuring additional manifests during cluster creation
- Creating a cluster on Amazon Web Services
- Creating a cluster on Microsoft Azure

- Creating a cluster on Google Cloud Platform
- Creating a cluster on VMware vSphere
- Creating a cluster on Red Hat OpenStack Platform
- Creating a cluster on Red Hat Virtualization
- Creating a cluster in an on-premises environment
- Creating a hosted cluster

1.4.4.1. Creating a cluster with the CLI

The multicluster engine for Kubernetes uses internal Hive components to create Red Hat OpenShift Container Platform clusters. See the following information to learn how to create clusters.

- Prerequisites
- Create a cluster with ClusterDeployment
- Create a cluster with cluster pool

1.4.4.1.1. Prerequisites

Before creating a cluster, you must clone the clusterImageSets repository and apply it to your hub cluster. See the following steps:

1. Run the following command to clone:

```
git clone https://github.com/stolostron/acm-hive-openshift-releases.git
cd acm-hive-openshift-releases
git checkout origin/release-2.6
```

2. Run the following command to apply it to your hub cluster:

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

Select the Red Hat OpenShift Container Platform release images when you create a cluster.

1.4.4.1.2. Create a cluster with ClusterDeployment

A **ClusterDeployment** is a Hive custom resource. See the following documentation to learn how to create an individual cluster:

Follow the Using Hive documentation to create the **ClusterDeployment** custom resource.

1.4.4.1.3. Create a cluster with ClusterPool

A **ClusterPool** is also a Hive custom resource that is used to create multiple clusters. Create a cluster with the Hive **ClusterPool** API.

Follow the Cluster Pools documentation to provision a cluster.

1.4.4.2. Configuring additional manifests during cluster creation

You can configure additional Kubernetes resource manifests during the installation process of creating your cluster. This can help if you need to configure additional manifests for scenarios such as configuring networking or setting up a load balancer.

Before you create your cluster, you need to add a reference to the **ClusterDeployment** resource that specifies a **ConfigMap** that contains the additional resource manifests.

Note: The **ClusterDeployment** resource and the **ConfigMap** must be in the same namespace. The following examples show how your content might look.

- ConfigMap with resource manifests
  ConfigMap that contains a manifest with another **ConfigMap** resource. The resource manifest **ConfigMap** can contain multiple keys with resource configurations added in a **data.<resource_name>\.yaml** pattern.

```
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"
```

- ClusterDeployment with resource manifest **ConfigMap** referenced
  The resource manifest **ConfigMap** is referenced under **spec.provisioning.manifestsConfigMapRef**.

```
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>
```

1.4.4.3. Creating a cluster on Amazon Web Services

You can use the multicluster engine for Kubernetes operator console to create a Red Hat OpenShift Container Platform cluster on Amazon Web Services (AWS).

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on AWS in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.4.4.3.1. Prerequisites

See the following prerequisites before creating a cluster on AWS:

- You must have a deployed multicluster engine for Kubernetes operator hub cluster.
- You need Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on Amazon Web Services.
- You need an AWS credential. See Creating a credential for Amazon Web Services for more information.
- You need a configured domain in AWS. See Configuring an AWS account for instructions on how to configure a domain.

- You must have Amazon Web Services (AWS) login credentials, which include user name, password, access key ID, and secret access key. See Understanding and Getting Your Security Credentials.

- You must have an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key on the cloud provider, you also need to manually update the corresponding credential for the cloud provider on the console of multicluster engine for Kubernetes operator. This is required when your credentials expire on the cloud provider where the managed cluster is hosted and you try to delete the managed cluster.

1.4.4.3.2. Creating your cluster with the console

To create a cluster from the console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Amazon Web Services for more information.

The name of the cluster is used in the hostname of the cluster.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.3.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured with your AWS account, that value is populated in the field. You can change the value by overwriting it. This name is used in the hostname of the cluster. See Configuring an AWS account for more information.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

- Zones: Specify where you want to run your control plane pools. You can select multiple zones within the region for a more distributed group of control plane nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.

- Instance type: Specify the instance type for your control plane node. You can change the type and size of your instance after it is created.

- Root storage: Specify the amount of root storage to allocate for the cluster.

You can create zero or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The optional information includes the following fields:

- Zones: Specify where you want to run your worker pools. You can select multiple zones within the region for a more distributed group of nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.

- Instance type: Specify the instance type of your worker pools. You can change the type and size of your instance after it is created.

- Node count: Specify the node count of your worker pool. This setting is required when you define a worker pool.

- Root storage: Specify the amount of root storage allocated for your worker pool. This setting is required when you define a worker pool.

Networking details are required for your cluster, and multiple networks are required for using IPv6. You can add an additional network by clicking **Add network**.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.

- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.

- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.

- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.4. Creating a cluster on Microsoft Azure

You can use the multicluster engine for Kubernetes operator console to deploy a Red Hat OpenShift Container Platform cluster on Microsoft Azure or on Microsoft Azure Government.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on Azure in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.4.4.4.1. Prerequisites

See the following prerequisites before creating a cluster on Azure:

- You must have a deployed multicluster engine for Kubernetes operator hub cluster.
- You need Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on Azure or Azure Government
- You need an Azure credential. See Creating a credential for Microsoft Azure for more information.
- You need a configured domain in Azure or Azure Government. See Configuring a custom domain name for an Azure cloud service for instructions on how to configure a domain.
- You need Azure login credentials, which include user name and password. See the Microsoft Azure Portal.
- You need Azure service principals, which include **clientId**, **clientSecret**, and **tenantId**. See azure.microsoft.com.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key on the cloud provider, you also need to manually update the corresponding credential for the cloud provider on the console of multicluster engine for Kubernetes operator. This is required when your credentials expire on the cloud provider where the managed cluster is hosted and you try to delete the managed cluster.

1.4.4.4.2. Creating your cluster with the console

To create a cluster from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Microsoft Azure for more information.

The name of the cluster is used in the hostname of the cluster.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.4.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Azure account, that value is populated in that field. You can change the value by overwriting it. See Configuring a custom domain name for an Azure cloud service for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The Node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following optional fields:

- Region: Specify a region where you want to run your node pools. You can select multiple zones within the region for a more distributed group of control plane nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Instance type and Root storage allocation (required) for your control plane pool. You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Zones: Specifies here you want to run your worker pools. You can select multiple zones within the region for a more distributed group of nodes. A closer zone might provide faster performance, but a more distant zone might be more distributed.
- Instance type: You can change the type and size of your instance after it is created.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.5. Creating a cluster on Google Cloud Platform

Follow the procedure to create a Red Hat OpenShift Container Platform cluster on Google Cloud Platform (GCP). For more information about GCP, see Google Cloud Platform.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on GCP in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.4.4.5.1. Prerequisites

See the following prerequisites before creating a cluster on GCP:

- You must have a deployed multicluster engine for Kubernetes operator hub cluster.
- You need Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on GCP.
- You must have a GCP credential. See Creating a credential for Google Cloud Platform for more information.
- You must have a configured domain in GCP. See Setting up a custom domain for instructions on how to configure a domain.
- You need your GCP login credentials, which include user name and password.

- You must have an OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: If you change your cloud provider access key on the cloud provider, you also need to manually update the corresponding credential for the cloud provider on the console of multicluster engine for Kubernetes operator. This is required when your credentials expire on the cloud provider where the managed cluster is hosted and you try to delete the managed cluster.

1.4.4.5.2. Creating your cluster with the console

To create clusters from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Google Cloud Platform for more information.

The name of your cluster is used in the hostname of the cluster. There are some restrictions that apply to naming your GCP cluster. These restrictions include not beginning the name with **goog** or containing a group of letters and numbers that resemble **google** anywhere in the name. See Bucket naming guidelines for the complete list of restrictions.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.5.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential for your GCP account, that value is populated in the field. You can change the value by overwriting it. See Setting up a custom domain for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The Node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- Region: Specify a region where you want to run your control plane pools. A closer region might provide faster performance, but a more distant region might be more distributed.

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

- Instance type: You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Instance type: You can change the type and size of your instance after it is created.

- Node count: This setting is required when you define a worker pool.

The networking details are required, and multiple networks are required for using IPv6 addresses. You can add an additional network by clicking **Add network**.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.

- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.

- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.

- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.6. Creating a cluster on VMware vSphere

You can use the multicluster engine for Kubernetes operator console to deploy a Red Hat OpenShift Container Platform cluster on VMware vSphere.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on vSphere in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites

- Creating your cluster with the console

- Adding your cluster to an existing cluster set

1.4.4.6.1. Prerequisites

See the following prerequisites before creating a cluster on vSphere:

- You must have a hub cluster that is deployed on OpenShift Container Platform version 4.6 or later.
- You need Internet access for your hub cluster so it can create the Kubernetes cluster on vSphere.
- You need a vSphere credential. See Creating a credential for VMware vSphere for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.
- You must have the following information for the VMware instance where you are deploying:
  - Required static IP addresses for API and Ingress instances
  - DNS records for:
    - **api.<cluster_name>.<base_domain>** which must point to the static API VIP
    - ***.apps.<cluster_name>.<base_domain>** which must point to the static IP address for Ingress VIP

Note: When creating a cluster by using the VMware vSphere or Red Hat OpenStack Platform providers and disconnected installation, if a certificate is required to access the mirror registry, you must enter it in the *Additional trust bundle* field of your credential in the *Configuration for disconnected installation section*. You cannot enter them in the cluster creation console editor.

1.4.4.6.2. Creating your cluster with the console

To create a cluster from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for VMware vSphere for more information about creating a credential.

The name of your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.6.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct

permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base domain associated with the selected credential that you configured for your vSphere account, that value is populated in the field. You can change the value by overwriting it. See Installing a cluster on vSphere with customizations for more information. This value must match the name that you used to create the DNS records listed in the prerequisites section. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images

Note: Only release images for OpenShift Container Platform versions 4.5.x and higher are supported.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the *Architecture* field. View the following field description:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes *Cores per socket*, *CPUs*, *Memory_min MB, _Disk size* in GiB, and *Node count*.

Networking information is required. Multiple networks are required for using IPv6. Some of the required networking information is included the following fields:

- vSphere network name: Specify the VMware vSphere network name.
- API VIP: Specify the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.
- Ingress VIP: Specify the IP address to use for ingress traffic.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.

- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Provide a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

You can define the disconnected installation image by clicking **Disconnected installation**. When creating a cluster by using the VMware vSphere provider and disconnected installation, if a certificate is required to access the mirror registry, you must enter it in the *Additional trust bundle* field of your credential in the *Configuration for disconnected installation section* . If you enter that certificate in the cluster create console editor, it is ignored.

You can click **Add automation template** to create a template.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.7. Creating a cluster on Red Hat OpenStack Platform

You can use the multicluster engine for Kubernetes operator console to deploy a Red Hat OpenShift Container Platform cluster on Red Hat OpenStack Platform.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on OpenStack in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.4.4.7.1. Prerequisites

See the following prerequisites before creating a cluster on Red Hat OpenStack Platform:

- You must have a hub cluster that is deployed on OpenShift Container Platform version 4.6, or later.
- You need Internet access for your hub cluster so it can create the Kubernetes cluster on Red Hat OpenStack Platform.
- You must have a Red Hat OpenStack Platform credential. See Creating a credential for Red Hat OpenStack Platform for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets.

- You need the following information for the Red Hat OpenStack Platform instance where you are deploying:
  - Flavor name for the control plane and worker instances; for example, **m1.xlarge**
  - Network name for the external network to provide the floating IP addresses
  - Required floating IP addresses for API and ingress instances
  - DNS records for:
    - **api.<cluster_name>.<base_domain>**, which must point to the floating IP address for the API
    - ***.apps.<cluster_name>.<base_domain>**, which must point to the floating IP address for ingress

1.4.4.7.2. Creating your cluster with the console

To create a cluster from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Red Hat OpenStack Platform for more information.

The name of the cluster is used in the hostname of the cluster. The name must contain fewer than 15 characters. This value must match the name that you used to create the DNS records listed in the credential prerequisites section.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.7.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Red Hat OpenStack Platform account, that value is populated in the field. You can change the value by overwriting it. See Managing domains in the Red Hat OpenStack Platform documentation for more information. This name is used in the hostname of the cluster.

The release image identifies the version of the OpenShift Container Platform image that is used to

create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images. Only release images for OpenShift Container Platform versions 4.6.x and higher are supported.

The node pools include the control plane pool and the worker pools. The control plane nodes share the management of the cluster activity. The information includes the following fields:

- **Optional** Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.
- Instance type for your control plane pool: You can change the type and size of your instance after it is created.

You can create one or more worker nodes in a worker pool to run the container workloads for the cluster. They can be in a single worker pool, or distributed across multiple worker pools. If zero worker nodes are specified, the control plane nodes also function as worker nodes. The information includes the following fields:

- Instance type: You can change the type and size of your instance after it is created.
- Node count: Specify the node count for your worker pool. This setting is required when you define a worker pool.

Networking details are required for your cluster. You must provide the values for one or more networks for an IPv4 network. For an IPv6 network, you must define more than one network.

You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Define a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

You can define the disconnected installation image by clicking **Disconnected installation**. When creating a cluster by using Red Hat OpenStack Platform provider and disconnected installation, if a certificate is required to access the mirror registry, you must enter it in the *Additional trust bundle* field of your credential in the *Configuration for disconnected installation section* . If you enter that certificate in the cluster create console editor, it is ignored.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

When creating a cluster that uses an internal certificate authority (CA), you need to customize the YAML file for your cluster by completing the following steps:

1. With the **YAML** switch on at the review step, insert a **Secret** object at the top of the list with the CA certificate bundle. **Note:** If the Red Hat OpenStack Platform environment provides services using certificates signed by multiple authorities, the bundle must include the certificates to validate all of the required endpoints. The addition for a cluster named **ocp3** resembles the following example:

```
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. Modify the Hive **ClusterDeployment** object to specify the value of **certificatesSecretRef** in **spec.platform.openstack**, similar to the following example:

```
platform:
  openstack:
    certificatesSecretRef:
      name: ocp3-openstack-trust
    credentialsSecretRef:
      name: ocp3-openstack-creds
    cloud: openstack
```

   The previous example assumes that the cloud name in the **clouds.yaml** file is **openstack**.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.8. Creating a cluster on Red Hat Virtualization

You can use the multicluster engine for Kubernetes operator console to create a Red Hat OpenShift Container Platform cluster on Red Hat Virtualization.

When you create a cluster, the creation process uses the OpenShift Container Platform installer with the Hive resource. If you have questions about cluster creation after completing this procedure, see Installing on RHV in the OpenShift Container Platform documentation for more information about the process.

- Prerequisites
- Creating your cluster with the console
- Adding your cluster to an existing cluster set

1.4.4.8.1. Prerequisites

See the following prerequisites before creating a cluster on Red Hat Virtualization:

- You must have a deployed multicluster engine for Kubernetes operator hub cluster.
- You need Internet access for your multicluster engine for Kubernetes operator hub cluster so it can create the Kubernetes cluster on Red Hat Virtualization.
- You need a Red Hat Virtualization credential. See Creating a credential for Red Hat Virtualization for more information.
- You need a configured domain and virtual machine proxy for the oVirt Engine virtual machines. See Installing on RHV in the Red Hat OpenShift Container Platform documentation for instructions on how to configure a domain.
- You must have Red Hat Virtualization login credentials, which include your Red Hat Customer Portal username and password.
- You need an OpenShift Container Platform image pull secret. You can download your pull secret from: Pull secret. See Using image pull secrets for more information about pull secrets.

Note: If you change your cloud provider access key on the cloud provider, you also need to manually update the corresponding credential for the cloud provider on the console of multicluster engine for Kubernetes operator. This is required when your credentials expire on the cloud provider where the managed cluster is hosted and you try to delete the managed cluster.

1.4.4.8.2. Creating your cluster with the console

To create a cluster from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

If you need to create a credential, see Creating a credential for Red Hat Virtualization for more information.

The name of your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

1.4.4.8.3. Adding your cluster to an existing cluster set

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your Red Hat Virtualization account, that value is populated in that field. You can overwrite the value to change it.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

The information for your node pools includes the number of Cores, Sockets, Memory, and Disk size for the the control plane pool. The three control plane nodes share the management of the cluster activity. The information includes the *Architecture* field. View the following field description:

- Architecture: If the architecture type of the managed cluster is not the same as the architecture of your hub cluster, enter a value for the instruction set architecture of the machines in the pool. Valid values are *amd64*, *ppc64le*, *s390x*, and *arm64*.

The worker pool information requires the pool name, number of cores, memory allocation, disk size allocation, and node count for your worker pools. Your worker nodes within the worker pool can be in a single worker pool, or distributed across multiple worker pools.

The following networking details are required from your preconfigured oVirt environment.

- oVirt network name
- API VIP: Specify the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.
- Ingress VIP: Specify the IP address to use for ingress traffic.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.
- Network type: The default value is **OpenShiftSDN**. OVNKubernetes is the required setting for using IPv6.
- Cluster network CIDR: This is a number and list of IP addresses that can be used for the pod IP addresses. This block must not overlap another network block. The default value is **10.128.0.0/14**.
- Network host prefix: Set the subnet prefix length for each node. The default value is **23**.
- Service network CIDR: Provide a block of IP addresses for services. This block must not overlap another network block. The default value is **172.30.0.0/16**.
- Machine CIDR: Provide a block of IP addresses that are used by the OpenShift Container Platform hosts. This block must not overlap another network block. The default value is **10.0.0.0/16**.
  You can add an additional network by clicking **Add network**. You must have more than one network if you are using IPv6 addresses.

Proxy information that is provided in the credential is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: Specify the URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: Specify the secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: Provide a comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.
- Additional trust bundle: Specify the contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can click the **YAML** switch **On** to view the **install-config.yaml** file content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You do not have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is automatically configured under the management of multicluster engine for Kubernetes operator.

1.4.4.9. Creating a cluster in an on-premises environment

You can use the console to create an on-premises Red Hat OpenShift Container Platform cluster. Use this process instead of the bare metal process that is deprecated.

You can use this procedure to create single-node OpenShift (SNO) clusters, multi-node clusters, and compact three-node clusters on VMware vSphere, Red Hat OpenStack, Red Hat Virtualization Platform, and in a bare metal environment.

You can also provision multiple single-node OpenShift clusters on edge resources by using the zero touch provisioning feature, which is a feature that is available with Red Hat OpenShift Container Platform. For more information, see Deploying distributed units at scale in a disconnected environment in the OpenShift Container Platform documentation.

- Prerequisites
- Creating your cluster with the console

1.4.4.9.1. Prerequisites

See the following prerequisites before creating a cluster in an on-premises environment:

- You must have a deployed hub cluster on OpenShift Container Platform version 4.9, or later.
- You need a configured infrastructure environment with a host inventory of configured hosts.
- You must have internet access for your hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the internet (disconnected) to retrieve the required images for creating the cluster.
- You need a configured on-premises credential.

- You need an OpenShift Container Platform image pull secret. See Using image pull secrets for more information.

1.4.4.9.2. Creating your cluster with the console

To create a cluster from the console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** and complete the steps in the console.

- Select **Host inventory** as the type of cluster.

The following options are available for your assisted installation:

- **Use existing discovered hosts**: Select your hosts from a list of hosts that are in an existing host inventory.
- **Discover new hosts**: Discover hosts that are not already in an existing infrastructure environment. Discover your own hosts, rather than using one that is already in an infrastructure environment.

If you need to create a credential, see Creating a credential for an on-premises environment for more information.

The name for your cluster is used in the hostname of the cluster.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If there is already a base DNS domain that is associated with the selected credential that you configured for your provider account, that value is populated in that field. You can change the value by overwriting it, but this setting cannot be changed after the cluster is created. The base domain of your provider is used to create routes to your Red Hat OpenShift Container Platform cluster components. It is configured in the DNS of your cluster provider as a Start of Authority (SOA) record.

The **OpenShift version** identifies the version of the OpenShift Container Platform image that is used to create the cluster. If the version that you want to use is available, you can select the image from the list of images. If the image that you want to use is not a standard image, you can enter the URL to the image that you want to use. See Release images for more information about release images.

When you select an OpenShift version that is 4.9 or later, an option to select **Install single node OpenShift (SNO)** is displayed. A single-node OpenShift cluster contains a single node which hosts the control plane services and the user workloads. See Scaling hosts to an infrastructure environment to learn more about adding nodes to a single-node OpenShift cluster after it is created.

If you want your cluster to be a single-node OpenShift cluster, select the **single-node OpenShift** option. You can add additional workers to single-node OpenShift clusters by completing the following steps:

1. From the console, navigate to **Infrastructure** > **Clusters** and select the name of the cluster that you created or want to access.
2. Select **Actions** > **Add hosts** to add additional workers.

Note: The single-node OpenShift control plane requires 8 CPU cores, while a control plane node for a multinode control plane cluster only requires 4 CPU cores.

After you review and save the cluster, your cluster is saved as a draft cluster. You can close the creation process and finish the process later by selecting the cluster name on the *Clusters* page.

If you are using existing hosts, select whether you want to select the hosts yourself, or if you want them to be selected automatically. The number of hosts is based on the number of nodes that you selected. For example, a SNO cluster only requires one host, while a standard three-node cluster requires three hosts.

The locations of the available hosts that meet the requirements for this cluster are displayed in the list of *Host locations*. For distribution of the hosts and a more high-availability configuration, select multiple locations.

If you are discovering new hosts with no existing infrastructure environment, complete the steps in Scaling hosts to an infrastructure environment beginning with step 4 to define your hosts.

After the hosts are bound, and the validations pass, complete the networking information for your cluster by adding the following IP addresses:

- API VIP: Specifies the IP address to use for internal API communication.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **api.** resolves correctly.
- Ingress VIP: Specifies the IP address to use for ingress traffic.
  Note: This value must match the name that you used to create the DNS records listed in the prerequisites section. If not provided, the DNS must be pre-configured so that **test.apps.** resolves correctly.

You can view the status of the installation on the *Clusters* navigation page.

1.4.4.10. Creating a hosted cluster (Technology Preview)

You can use the multicluster engine for Kubernetes operator console to create a Red Hat OpenShift Container Platform hosted cluster.

- Prerequisites
- Creating a hosted cluster

1.4.4.10.1. Prerequisites

See the following prerequisites before creating a hosted cluster:

- You must have a deployed multicluster engine for Kubernetes operator hub cluster on OpenShift Container Platform version 4.6 or later.

- You need Internet access for your multicluster engine for Kubernetes operator hub cluster (connected), or a connection to an internal or mirror registry that has a connection to the Internet (disconnected) to retrieve the required images for creating the cluster.
- You must enable the hosted control plane operator. See Using hosted control plane clusters for more information.
- You need an OpenShift Container Platform image pull secret. See Using image pull secrets for more information.
- You need an infrastructure environment with discovered hosts.
  Note: When you create a cluster by using a host from your inventory and a disconnected installation, you must store all your settings in the credential in the *Configuration for disconnected installation* section. You cannot enter them in the cluster creation console editor.

1.4.4.10.2. Creating a hosted cluster

To create a cluster from the multicluster engine for Kubernetes operator console, navigate to **Infrastructure** > **Clusters**. On the *Clusters* page, click **Create cluster** > **Host inventory** > **Hosted** and complete the steps in the console.

Note: This procedure is for creating a cluster. If you have an existing cluster that you want to import, see Importing a target managed cluster to the hub cluster for those steps.

Important: When you create a cluster, the multicluster engine for Kubernetes operator controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Tip: Select **YAML: On** to view content updates as you enter the information in the console.

If you want to add your cluster to an existing cluster set, you must have the correct permissions on the cluster set to add it. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have any cluster set options to select.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

The release image identifies the version of the OpenShift Container Platform image that is used to create the cluster. Hosted clusters must use one of the provided release images.

Proxy information that is provided in the infrastructure environment is automatically added to the proxy fields. You can use the information as it is, overwrite it, or add the information if you want to enable a proxy. The following list contains the required information for creating a proxy:

- HTTP proxy URL: The URL that should be used as a proxy for **HTTP** traffic.
- HTTPS proxy URL: The secure proxy URL that should be used for **HTTPS** traffic. If no value is provided, the same value as the **HTTP Proxy URL** is used for both **HTTP** and **HTTPS**.
- No proxy domains: A comma-separated list of domains that should bypass the proxy. Begin a domain name with a period **.** to include all of the subdomains that are in that domain. Add an asterisk ***** to bypass the proxy for all destinations.

- Additional trust bundle: The contents of the certificate file that is required to access the mirror registry.

When you review your information and optionally customize it before creating the cluster, you can select **YAML: On** to view the YAML content in the panel. You can edit the YAML file with your custom settings, if you have any updates.

Note: You have to run the **kubectl** command that is provided with the cluster details to import the cluster. When you create the cluster, it is not automatically configured with the management of Red Hat Advanced Cluster Management.

1.4.4.11. Creating a cluster in a proxy environment

You can create a Red Hat OpenShift Container Platform cluster when your hub cluster is connected through a proxy server.

One of the following situations must be true for the cluster creation to succeed:

- multicluster engine for Kubernetes operator has a private network connection with the managed cluster that you are creating, but the Red Hat Advanced Cluster Management and managed cluster access the Internet using a proxy.
- The managed cluster is on a infrastructure provider, but the firewall ports enable communication from the managed cluster to the hub cluster.

To create a cluster that is configured with a proxy, complete the following steps:

1. Configure the cluster-wide-proxy setting on the hub cluster by adding the following information to your **install-config.yaml** file:

```
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>
```

   Replace **username** with the username for your proxy server.

   Replace **password** with the password to access your proxy server.

   Replace **proxy.example.com** with the path of your proxy server.

   Replace **port** with the communication port with the proxy server.

   Replace **wildcard-of-domain** with an entry for domains that should bypass the proxy.

   Replace **provisioning-network/CIDR** with the IP address of the provisioning network and the number of assigned IP addresses, in CIDR notation.

   Replace **BMC-address-range/CIDR** with the BMC address and the number of addresses, in CIDR notation.

   After you add the previous values, the settings are applied to your clusters.

2. Provision the cluster by completing the procedure for creating a cluster. See Creating a cluster to select your provider.

Note: You can only use **install-config** YAML when deploying your cluster. After deploying your cluster, any new changes you make to **install-config.yaml** do not apply. To update the configuration after deployment, you must policies. See Pod policy for more information.

1.4.4.11.1. Enabling cluster-wide proxy on existing cluster add-ons

You can configure the **KlusterletAddonConfig** in the cluster namespace to add the proxy environment variables to all of the klusterlet add-on pods of the Red Hat OpenShift Container Platform clusters that are managed by the hub cluster.

Complete the following steps to configure the **KlusterletAddonConfig** to add the 3 environment variables to the pods of the klusterlet add-ons:

1. Open the **KlusterletAddonConfig** file that is in the namespace of the cluster that needs the proxy added.

2. Edit the **.spec.proxyConfig** section of the file so it resembles the following example:

```
spec
  proxyConfig:
    httpProxy: "<proxy_not_secure>"
    httpsProxy: "<proxy_secure>"
    noProxy: "<no_proxy>"
```

Replace **proxy_not_secure** with the address of the proxy server for **http** requests. For example, **http://192.168.123.145:3128**.

Replace **proxy_secure** with the address of the proxy server for **https** requests. For example, **https://192.168.123.145:3128**.

Replace **no_proxy** with a comma delimited list of IP addresses, hostnames, and domain names where traffic will not be routed through the proxy. For example, **.cluster.local,.svc,10.128.0.0/14,example.com**.

The **spec.proxyConfig** is an optional section. If the OpenShift Container Platform cluster is created with cluster wide proxy configured on the hub cluster, the cluster wide proxy configuration values are added to the pods of the klusterlet add-ons as environment variables when the following conditions are met:

- The **.spec.policyController.proxyPolicy** in the **addon** section is enabled and set to **OCPGlobalProxy**

- The **.spec.applicationManager.proxyPolocy** is enabled and set to **CustomProxy**. **Note:** The default value of **proxyPolicy** in the **addon** section is **Disabled**.

  See the following example:

```
apiVersion: agent.open-cluster-management.io/v1
   kind: KlusterletAddonConfig
   metadata:
     name: clusterName
     namespace: clusterName
   spec:
```

```
proxyConfig:
  httpProxy: http://pxuser:12345@10.0.81.15:3128
  httpsProxy: http://pxuser:12345@10.0.81.15:3128
  noProxy: .cluster.local,.svc,10.128.0.0/14, example.com
applicationManager:
  enabled: true
  proxyPolicy: CustomProxy
policyController:
  enabled: true
  proxyPolicy: OCPGlobalProxy
searchCollector:
  enabled: true
  proxyPolicy: Disabled
certPolicyController:
  enabled: true
  proxyPolicy: Disabled
iamPolicyController:
  enabled: true
  proxyPolicy: Disabled
```

Important: Global proxy settings do not impact alert forwarding. To set up alert forwarding for Red Hat Advanced Cluster Management hub clusters with a cluster-wide proxy, see Forwarding alerts for more details.

1.4.5. Hibernating a created cluster (Technology Preview)

You can hibernate a cluster that was created using multicluster engine for Kubernetes operator to conserve resources. A hibernating cluster requires significantly fewer resources than one that is running, so you can potentially lower your provider costs by moving clusters in and out of a hibernating state. This feature only applies to clusters that were created by multicluster engine for Kubernetes operator in the following environments:

- Amazon Web Services
- Microsoft Azure
- Google Cloud Platform

1.4.5.1. Hibernate a cluster by using the console

To use the console to hibernate a cluster that was created by multicluster engine for Kubernetes operator, complete the following steps:

1. From the navigation menu, select **Infrastructure** > **Clusters**. Ensure that the *Manage clusters* tab is selected.
2. Select **Hibernate cluster** from the the *Options* menu for the cluster. **Note:** If the *Hibernate cluster* option is not available, you cannot hibernate the cluster. This can happen when the cluster is imported, and not created by multicluster engine for Kubernetes operator.

The status for the cluster on the *Clusters* page is **Hibernating** when the process completes.

Tip: You can hibernate multiple clusters by selecting the clusters that you want to hibernate on the *Clusters* page, and selecting **Actions** > **Hibernate clusters**.

Your selected cluster is hibernating.

1.4.5.2. Hibernate a cluster by using the CLI

To use the CLI to hibernate a cluster that was created by multicluster engine for Kubernetes operator, complete the following steps:

1. Enter the following command to edit the settings for the cluster that you want to hibernate:

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

2. Change the value for **spec.powerState** to **Hibernating**.

3. Enter the following command to view the status of the cluster:

   ```
   oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

   When the process of hibernating the cluster is complete, the value of the type for the cluster is **type=Hibernating**.

Your selected cluster is hibernating.

1.4.5.3. Resuming normal operation of a hibernating cluster by using the console

To resume normal operation of a hibernating cluster by using the console, complete the following steps:

1. From the navigation menu, select **Infrastructure** > **Clusters**. Ensure that the *Manage clusters* tab is selected.

2. Select **Resume cluster** from the the *Options* menu for the cluster that you want to resume.

The status for the cluster on the *Clusters* page is **Ready** when the process completes.

Tip: You can resume multiple clusters by selecting the clusters that you want to resume on the *Clusters* page, and selecting **Actions** > **Resume clusters**.

Your selected cluster is resuming normal operation.

1.4.5.4. Resuming normal operation of a hibernating cluster by using the CLI

To resume normal operation of a hibernating cluster by using the CLI, complete the following steps:

1. Enter the following command to edit the settings for the cluster:

   ```
   oc edit clusterdeployment <name-of-cluster> -n <namespace-of-cluster>
   ```

   Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

   Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

2. Change the value for **spec.powerState** to **Running**.

3. Enter the following command to view the status of the cluster:

```
oc get clusterdeployment <name-of-cluster> -n <namespace-of-cluster> -o yaml
```

Replace **name-of-cluster** with the name of the cluster that you want to hibernate.

Replace **namespace-of-cluster** with the namespace of the cluster that you want to hibernate.

When the process of resuming the cluster is complete, the value of the type for the cluster is **type=Running**.

Your selected cluster is resuming normal operation.

1.4.6. Importing a target managed cluster to the hub cluster

You can import clusters from different Kubernetes cloud providers. After you import, the targeted cluster becomes a managed cluster for the multicluster engine for Kubernetes operator hub cluster. Unless otherwise specified, complete the import tasks anywhere that you can access the hub cluster and the targeted managed cluster.

A hub cluster cannot manage *any* other hub cluster, but can manage itself. The hub cluster is configured to automatically be imported and self-managed. You do not need to manually import the hub cluster.

However, if you remove a hub cluster and try to import it again, you need to add the **local-cluster:true** label.

Choose from the following instructions to set up your managed cluster, either from the console or from the CLI:

Required user type or access level Cluster administrator

- Importing an existing cluster with the console
- Importing a managed cluster with the CLI

1.4.6.1. Importing an existing cluster with the console

After you install multicluster engine for Kubernetes operator, you are ready to import a cluster to manage. You can import from both the console and the CLI.

Follow this procedure to import from the console. You need your terminal for authentication during this procedure.

- Prerequisites
- Importing a cluster
- Removing a cluster

1.4.6.1.1. Prerequisites

- You need a hub cluster that is deployed. If you are importing bare metal clusters, you must have the hub cluster installed on Red Hat OpenShift Container Platform version 4.8 or later.

- You need a cluster that you want to manage and Internet connectivity.
- Install **kubectl**. To install **kubectl**, see *Install and Set Up kubectl* in the Kubernetes documentation.
- You need the **base64** command line tool.
- **Note:** If you are importing a cluster that was not created by OpenShift Container Platform, you need a **multiclusterhub.spec.imagePullSecret** defined. This secret might be created when multicluster engine for Kubernetes operator was installed.
  If you need to create a new secret, complete the following steps:

  1. Download your Kubernetes pull secret from cloud.redhat.com.
  2. Add the pull secret to the namespace of your hub cluster.
  3. Run the following command to create a new secret in the **open-cluster-management** namespace:

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

     Replace **open-cluster-management** with the name of the namespace of your hub cluster. The default namespace of the hub cluster is **open-cluster-management**.

     Replace **path-to-pull-secret** with the path to the pull secret that you downloaded.

     The secret is automatically copied to the managed cluster when it is imported.

     See Using image pull secrets, or Understanding and creating service accounts for more information about pull secrets.

     See Custom image pull secret for more information about how to define this secret.

- Ensure the agent is deleted on the cluster that you want to import. The **open-cluster-management-agent** and **open-cluster-management-agent-addon** namespaces must be removed to avoid errors.
- For importing in a Red Hat OpenShift Dedicated environment, see the following notes:
  - You must have the hub cluster deployed in a Red Hat OpenShift Dedicated environment.
  - The default permission in Red Hat OpenShift Dedicated is dedicated-admin, but that does not contain all of the permissions to create a namespace. You must have **cluster-admin** permissions to import and manage a cluster with multicluster engine for Kubernetes operator.
- If you want to enable automation on your imported cluster, you need the Red Hat Ansible Automation Platform.

Required user type or access level Cluster administrator

1.4.6.1.2. Importing a cluster

You can import existing clusters from the console for each of the available cloud providers.

Note: A hub cluster cannot manage a different hub cluster. A hub cluster is set up to automatically import and manage itself, so you do not have to manually import a hub cluster to manage itself.

By default, the namespace is used for the cluster name and namespace, but you can change it.

Important: When you create a cluster, the controller creates a namespace for the cluster and its resources. Ensure that you include only resources for that cluster instance in that namespace. Destroying the cluster deletes the namespace and all of the resources in it.

Every managed cluster must be associated with a managed cluster set. If you do not assign the managed cluster to a **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

If you want to add it to a different cluster set, you must have **cluster-admin** privileges to the cluster set. If you do not have **cluster-admin** privileges when you are creating the cluster, you must select a cluster set on which you have **clusterset-admin** permissions. If you do not have the correct permissions on the specified cluster set, the cluster creation fails. Contact your cluster administrator to provide you with **clusterset-admin** permissions to a cluster set if you do not have cluster set options to select.

If you import a Red Hat OpenShift Dedicated cluster and do not specify a vendor by adding a label for **vendor=OpenShiftDedicated**, or if you add a label for **vendor=auto-detect**, a **managed-by=platform** label is automatically added to the cluster. You can use this added label to identify the cluster as a Red Hat OpenShift Dedicated cluster and retrieve the Red Hat OpenShift Dedicated clusters as a group.

The following list provides the available options for *import mode*, which specifies the method for importing the cluster:

- Run import commands manually: After completing and submitting the information in the console, including any Red Hat Ansible Automation Platform templates, run the provided command on the target cluster to import the cluster. You must have the Red Hat Ansible Automation Platform Resource Operator installed from OperatorHub to create and run an Ansible Automation Platform job. Continue with the steps in Additional steps for running the import commands manually.

- Enter your server URL and API token for the existing cluster: Provide the server URL and API token of the cluster that you are importing.
  You can specify a Red Hat Ansible Automation Platform template to run when the cluster is upgraded. You must have the Red Hat Ansible Automation Platform Resource Operator installed from OperatorHub to create and run an Ansible Automation Platform job.

  If you want to configure a cluster API address, complete the steps in Optional: Configure the cluster API address.

- Provide the **kubeconfig** file: Copy and paste the contents of the **kubeconfig** file of the cluster that you are importing.
  You can specify a Red Hat Ansible Automation Platform template to run when the cluster is upgraded. You must have the Red Hat Ansible Automation Platform Resource Operator installed from OperatorHub to create and run an Ansible Automation Platform job.

  If you want to configure a cluster API address, continue with the steps in Optional: Configure the cluster API address.

1.4.6.1.2.1. Additional steps for running the import commands manually

Run the provided command on the target cluster to import the cluster.

After submitting the cluster details and automation information, select **Copy command** to copy the generated command and token to the clipboard.

Important: The command contains pull secret information that is copied to each of the imported clusters. Anyone who can access the imported clusters can also view the pull secret information. Consider creating a secondary pull secret at https://cloud.redhat.com/ or create a service account to protect your personal credentials.

1. Log in to the managed cluster that you want to import.
2. For the Red Hat OpenShift Dedicated environment only, complete the following steps (if you are not using a Red Hat OpenShift Dedicated environmnet, skip to step 3):
   a. Create the **open-cluster-management-agent** and **open-cluster-management** namespaces or projects on the managed cluster.
   b. Find the klusterlet operator in the OpenShift Container Platform catalog.
   c. Install it in the **open-cluster-management** namespace or project that you created. **Important:** Do not install the operator in the **open-cluster-management-agent** namespace.
   d. Extract the bootstrap secret from the import command by completing the following steps:
      i. Generate the import command:
         A. Select **Infrastructure** > **Clusters** from the console navigation.
         B. Select **Add a cluster** > **Import an existing cluster**.
         C. Add the cluster information, and select **Save import and generate code**
      ii. Copy the import command.
      iii. Paste the import command into a file that you create named **import-command**.
      iv. Run the following command to insert the content into the new file:

```
cat import-command | awk '{split($0,a,"&&"); print a[3]}' | awk '{split($0,a,"|"); print a[1]}' | sed -e "s/^ echo //" | base64 -d
```

      v. Find and copy the secret with the name **bootstrap-hub-kubeconfig** in the output.
      vi. Apply the secret to the **open-cluster-management-agent** namespace on the managed cluster.
      vii. Create the klusterlet resource using the example in the installed operator, the clusterName should be changed the same name as cluster name that was set during the import.
      Note: When the **managedcluster** resource is successfully registered to the hub, there are two klusterlet operators installed. One klusterlet operator is in the **open-cluster-management** namespace, and the other is in the **open-cluster-management-agent** namespace. Multiple operators does not affect the function of the klusterlet.
3. For cluster imports that are not in the Red Hat OpenShift Dedicated environment, complete the following steps:
   a. If necessary, configure your **kubectl** commands for your managed cluster.

b. To deploy the **open-cluster-management-agent-addon** to the managed cluster, run the command and token that you copied.

4. Select **View cluster** to view a summary of your cluster in the *Overview* page.
You can view the progress on the *Cluster details* page for the cluster as it is imported.

If you want to configure a cluster API address, continue with the steps in Optional: Configure the cluster API address.

1.4.6.1.2.2. Optional: Configure the cluster API address

You can optionally configure the **Cluster API address** that is on the cluster details page by configuring the URL that is displayed in the table when you run the **oc get managedcluster** command.

1. Log in to your hub cluster with an ID that has **cluster-admin** permissions.
2. Configure your **kubectl** for your targeted managed cluster.
3. Edit the managed cluster entry for the cluster that you are importing by entering the following command:

```
oc edit managedcluster <cluster-name>
```

Replace ***cluster-name*** with the name of the managed cluster.

4. Add the **ManagedClusterClientConfigs** section to the **ManagedCluster** spec in the YAML file, as shown in the following example:

```
spec:
  hubAcceptsClient: true
  managedClusterClientConfigs:
  - url: https://multicloud-console.apps.new-managed.dev.redhat.com
```

Replace the value of the URL with the URL that provides external access to the managed cluster that you are importing.

1.4.6.1.3. Removing an imported cluster

Complete the following procedure to remove an imported cluster and the **open-cluster-management-agent-addon** that was created on the managed cluster.

On the *Clusters* page, click **Actions** > **Detach cluster** to remove your cluster from management.

Note: If you attempt to detach the hub cluster, which is named **local-cluster**, be aware that the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport. If you want to reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**. For more information, see the disableHubSelfManagement topic.

1.4.6.2. Importing a managed cluster with the CLI

After you install multicluster engine for Kubernetes operator, you are ready to import a cluster to manage by using the Red Hat OpenShift Container Platform CLI. You can import a cluster using the **kubeconfig** file of the cluster that you are importing, or you can run import commands manually on the cluster you are importing. Both procedures are documented.

- Prerequisites
- Supported architecture
- Preparing for import
- Importing the cluster with the auto import secret
- Importing the cluster with the manual commands
- Importing the klusterlet add-on

Important: A hub cluster cannot manage a different hub cluster. A hub cluster is set up to automatically import and manage itself. You do not have to manually import a hub cluster to manage itself. However, if you remove a hub cluster and try to import it again, you need to add the **local-cluster:true** label.

1.4.6.2.1. Prerequisites

- You need a hub cluster that is deployed. If you are importing bare metal clusters, you must have the hub cluster installed on Red Hat OpenShift Container Platform version 4.6 or later.
- You need a separate cluster that you want to manage that has Internet connectivity.
- You need the Red Hat OpenShift Container Platform CLI version 4.6 or later, to run **oc** commands. See Getting started with the OpenShift CLI for information about installing and configuring the Red Hat OpenShift Container Platform CLI, **oc**. Download the installation file for CLI tools from the console.
- You need to install the Kubernetes CLI, **kubectl**. To install **kubectl**, see *Install and Set Up kubectl* in the Kubernetes documentation.
- If you are importing a cluster that was not created by OpenShift Container Platform, you need a **multiclusterhub.spec.imagePullSecret** defined. This secret might have been created when multicluster engine for Kubernetes operator was installed. See Custom image pull secret for more information about how to define this secret.

1.4.6.2.2. Supported architectures

- Linux (x86_64, s390x, ppc64le)
- macOS

1.4.6.2.3. Preparing for import

1. Log in to your *hub cluster* by running the following command:

```
oc login
```

2. Run the following command on the hub cluster to create the project and namespace. The cluster name that is defined in **CLUSTER_NAME** is also used as the cluster namespace in the YAML file and commands:

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

Important: The **cluster.open-cluster-management.io/managedCluster** label is automatically added to and removed from a managed cluster namespace. Do not manually add it to or remove it from a managed cluster.

3. Create a file named **managed-cluster.yaml** with the following example content:

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

When the values for **cloud** and **vendor** are set to **auto-detect**, Red Hat Advanced Cluster Management detects the cloud and vendor types automatically from the cluster that you are importing. You can optionally replace the values for **auto-detect** with with the cloud and vendor values for your cluster. See the following example:

```
cloud: Amazon
vendor: OpenShift
```

4. Apply the YAML file to the **ManagedCluster** resource by entering the following command:

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

Next, import the cluster by completing the steps in one of the following procedures:

- Importing the cluster with the auto import secret
- Importing the cluster with the manual commands

1.4.6.2.4. Importing the cluster with the auto import secret

To import with the auto import secret, you must create a secret that contains either a reference to the **kubeconfig** file of the cluster or the kube API server and token pair of the cluster.

1. Retrieve the **kubeconfig** file or the kube API server and token of the cluster that you are importing. See the documentation for your Kubernetes cluster to learn where to locate your **kubeconfig** file or your kube API server and token.

2. Create the **auto-import-secret.yaml** file in the ${CLUSTER_NAME} namespace.

a. Create a YAML file named **auto-import-secret.yaml** that contains content that is similar to the following template:

```
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. Apply the YAML file in the ${CLUSTER_NAME} namespace with the following command:

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

Note: By default, the auto-import secret is used one time and deleted when the import process completes. If you want to keep the auto import secret, add **managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret** to the secret. You can add it by running the following command:

```
oc -n <cluster_name> annotate secrets auto-import-secret managedcluster-import-controller.open-cluster-management.io/keeping-auto-import-secret=""
```

3. Validate the **JOINED** and **AVAILABLE** status for your imported cluster. Run the following command from the hub cluster:

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

4. Log in to the managed cluster by running the following command on the managed cluster:

```
oc login
```

5. Run the following command to validate the pod status on the cluster that you are importing:

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

Next, complete the steps in Importing the klusterlet add-on. Importing the add-on is a necessary step to finish the entire process of importing a cluster.

1.4.6.2.5. Importing the cluster with the manual commands

Important: The import command contains pull secret information that is copied to each of the imported clusters. Anyone who can access the imported clusters can also view the pull secret information.

1. Obtain the **klusterlet-crd.yaml** file that was generated by the import controller on your hub cluster by running the following command:

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

2. Obtain the **import.yaml** file that was generated by the import controller on your hub cluster by running the following command:

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

Proceed with the following steps in the cluster that you are importing:

3. Log in to the managed cluster that you are importing by entering the following command:

```
oc login
```

4. Apply the **klusterlet-crd.yaml** that you generated in step 1 by running the following command:

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

5. Apply the **import.yaml** file that you previously generated by running the following command:

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

6. Validate **JOINED** and **AVAILABLE** status for the cluster that you are importing. From the hub cluster, run the following command:

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

Next, complete the steps in Importing the klusterlet add-on. Importing the add-on is a necessary step to finish the entire process of importing a cluster.

1.4.6.2.6. Importing the klusterlet add-on

You can create and apply the klusterlet add-on configuration file by completing the following procedure:

1. Create a YAML file that is similar to the following example:

```
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. Save the file as **klusterlet-addon-config.yaml**.

3. Apply the YAML by running the following command:

```
oc apply -f klusterlet-addon-config.yaml
```

   The ManagedCluster-Import-Controller will generate a secret named **${CLUSTER_NAME}-import**. The **${CLUSTER_NAME}-import** secret contains the **import.yaml** that the user applies to a managed cluster to install klusterlet.

   Add-ons are installed after the cluster you are importing is **AVAILABLE**.

4. Validate the pod status of add-ons on the cluster you are importing by running the following command:

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

1.4.6.2.7. Removing an imported cluster with the CLI

To remove a cluster, run the following command:

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

Replace **cluster_name** with the name of the cluster.

Your cluster is now removed.

1.4.6.3. Importing a cluster with a custom ManagedClusterImageRegistry CRD

There might be times when you need to override the image registry on the managed clusters that you are importing. You can do this by creating a **ManagedClusterImageRegistry** custom resource definition (CRD).

The **ManagedClusterImageRegistry** CRD is a namespace-scoped resource.

The **ManagedClusterImageRegistry** CRD specifies a set of managed clusters for a Placement to select, but needs different images from the custom image registry. After the managed clusters are updated with the new images, the following label is added to each managed cluster for identification: **open-cluster-management.io/image-registry=<namespace>.<managedClusterImageRegistryName>**.

The following example shows a **ManagedClusterImageRegistry** CRD:

```
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>
  pullSecret:
    name: <pullSecretName>
  registries:
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
  - mirror: <mirrored-image-registry-address>
    source: <image-registry-address>
```

In the **spec** section:

- Replace **placementName** with the name of a Placement in the same namespace that selects a set of managed clusters.
- Replace **pullSecretName** with the name of the pull secret that is used to pull images from the custom image registry.
- List the values for each of the **source** and **mirror** registries. Replace the **mirrored-image-registry-address** and **image-registry-address** with the value for each of the **mirror** and **source** values of the registries.
  - Example 1: To replace the source image registry named **registry.redhat.io/rhacm2** with **localhost:5000/rhacm2**, and **registry.redhat.io/multicluster-engine** with **localhost:5000/multicluster-engine**, use the following example:

    ```
    registries:
    - mirror: localhost:5000/rhacm2/
        source: registry.redhat.io/rhacm2
    - mirror: localhost:5000/multicluster-engine
        source: registry.redhat.io/multicluster-engine
    ```

  - Example 2: To replace the source image, **registry.redhat.io/rhacm2/registration-rhel8-operator** with **localhost:5000/rhacm2-registration-rhel8-operator**, use the following example:

    ```
    registries:
    - mirror: localhost:5000/rhacm2-registration-rhel8-operator
        source: registry.redhat.io/rhacm2/registration-rhel8-operator
    ```

1.4.6.3.1. Importing a cluster with a ManagedClusterImageRegistry CRD

Complete the following steps to import a cluster with a ManagedClusterImageRegistry CRD:

1. Create a pull secret in the namespace where you want your cluster to be imported. For these steps, it is **myNamespace**.

   ```
   $ kubectl create secret docker-registry myPullSecret \
     --docker-server=<your-registry-server> \
     --docker-username=<my-name> \
     --docker-password=<my-password>
   ```

2. Create a Placement in the namespace that you created.

```
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
```

Note: The **unreachable** toleration is required for the Placement to be able to select the cluster.

3. Create a **ManagedClusterSet** resource and bind it to your namespace.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSet
metadata:
  name: myClusterSet

---
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSetBinding
metadata:
  name: myClusterSet
  namespace: myNamespace
spec:
  clusterSet: myClusterSet
```

4. Create the **ManagedClusterImageRegistry** CRD in your namespace.

```
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. Import a managed cluster from the console and add it to a managed cluster set.

6. Copy and run the import commands on the managed cluster after the label **open-cluster-management.io/image-registry=myNamespace.myImageRegistry** is added to the managed cluster.

1.4.7. Removing a cluster from management

When you remove an OpenShift Container Platform cluster from management that was created with

multicluster engine for Kubernetes operator, you can either *detach* it or *destroy* it. Detaching a cluster removes it from management, but does not completely delete it. You can import it again if you want to manage it. This is only an option when the cluster is in a *Ready* state.

The following procedures remove a cluster from management in either of the following situations:

- You already deleted the cluster and want to remove the deleted cluster from Red Hat Advanced Cluster Management.
- You want to remove the cluster from management, but have not deleted the cluster.

Important:

- Destroying a cluster removes it from management and deletes the components of the cluster.
- When you detach a managed cluster, the related namespace is automatically deleted. Do not place custom resources in this namespace.
  - Removing a cluster by using the console
  - Removing a cluster by using the command line
  - Removing remaining resources after removing a cluster
  - Defragmenting the etcd database after removing a cluster

1.4.7.1. Removing a cluster by using the console

From the navigation menu, navigate to **Infrastructure** > **Clusters** and select **Destroy cluster** or **Detach cluster** from the options menu beside the cluster that you want to remove from management.

+ **Tip:** You can detach or destroy multiple clusters by selecting the check boxes of the clusters that you want to detach or destroy and selecting **Detach** or **Destroy**.

Note: If you attempt to detach the hub cluster while it is managed, which is called a **local-cluster**, check to see if the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached, and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport.

To reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**, as described in Installing while connected online .

1.4.7.2. Removing a cluster by using the command line

To detach a managed cluster by using the command line of the hub cluster, run the following command:

```
oc delete managedcluster $CLUSTER_NAME
```

To delete the managed cluster after detaching, run the following command:

```
oc delete clusterdeployment <CLUSTER_NAME> -n $CLUSTER_NAME
```

Note: If you attempt to detach the hub cluster, which is named **local-cluster**, be aware that the default setting of **disableHubSelfManagement** is **false**. This setting causes the hub cluster to reimport itself and manage itself when it is detached and it reconciles the **MultiClusterHub** controller. It might take hours for the hub cluster to complete the detachment process and reimport. If you want to reimport the hub cluster without waiting for the processes to finish, you can enter the following command to restart the **multiclusterhub-operator** pod and reimport faster:

```
oc delete po -n open-cluster-management `oc get pod -n open-cluster-management | grep
multiclusterhub-operator| cut -d' ' -f1`
```

You can change the value of the hub cluster to not import automatically by changing the **disableHubSelfManagement** value to **true**, as described in Installing while connected online .

1.4.7.3. Removing remaining resources after removing a cluster

If there are remaining resources on the managed cluster that you removed, there are additional steps that are required to ensure that you remove all of the remaining components. Situations when these extra steps are required include the following examples:

- The managed cluster was detached before it was completely created, and components like the **klusterlet** remain on the managed cluster.
- The hub that was managing the cluster was lost or destroyed before detaching the managed cluster, and there is no way to detach the managed cluster from the hub.
- The managed cluster was not in an online state when it was detached.

If one of these situations apply to your attempted detachment of a managed cluster, there are some resources that cannot be removed from managed cluster. Complete the following steps to detach the managed cluster:

1. Make sure you have the **oc** command line interface configured.
2. Make sure you have **KUBECONFIG** configured on your managed cluster.
   If you run **oc get ns | grep open-cluster-management-agent**, you should see two namespaces:

   ```
   open-cluster-management-agent         Active   10m
   open-cluster-management-agent-addon   Active   10m
   ```

3. Run the following command to remove the remaining resources:

   ```
   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": []}}'
   ```

4. Run the following command to ensure that both namespaces and all open cluster management **crds** are removed:

   ```
   oc get crds | grep open-cluster-management.io | awk '{print $1}'
   oc get ns | grep open-cluster-management-agent
   ```

1.4.7.4. Defragmenting the etcd database after removing a cluster

Having many managed clusters can affect the size of the **etcd** database in the hub cluster. In OpenShift Container Platform 4.8, when you delete a managed cluster, the **etcd** database in the hub cluster is not automatically reduced in size. In some scenarios, the **etcd** database can run out of space. An error **etcdserver: mvcc: database space exceeded** is displayed. To correct this error, reduce the size of the **etcd** database by compacting the database history and defragmenting the **etcd** database.

Note: For OpenShift Container Platform version 4.9 and later, the etcd Operator automatically defragments disks and compacts the **etcd** history. No manual intervention is needed. The following procedure is for OpenShift Container Platform version 4.8 and earlier.

Compact the **etcd** history and defragment the **etcd** database in the hub cluster by completing the following procedure.

1.4.7.4.1. Prerequisites

- Install the OpenShift CLI (**oc**).
- Log in as a user with **cluster-admin** privileges.

1.4.7.4.2. Procedure

1. Compact the **etcd** history.
   a. Open a remote shell session to the **etcd** member, for example:

   ```
   $ oc rsh -n openshift-etcd etcd-control-plane-0.example.com etcdctl endpoint status --cluster -w table
   ```

   b. Run the following command to compact the **etcd** history:

   ```
   sh-4.4#etcdctl compact $(etcdctl endpoint status --write-out="json" |  egrep -o '"revision":[0-9]*' | egrep -o '[0-9]*' -m1)
   ```

   Example output

   ```
   $ compacted revision 158774421
   ```

2. Defragment the **etcd** database and clear any **NOSPACE** alarms as outlined in Defragmenting **etcd** data.

1.4.8. Scaling managed clusters

For clusters that you created, you can customize and resize your managed cluster specifications, such as virtual machine sizes and number of nodes. See the following options:

- Scaling with MachinePool (Technology Preview)
- Scaling hosts to an infrastructure environment

1.4.8.1. Scaling with MachinePool (Technology Preview)

For clusters you created, you can customize and resize your managed cluster specifications, such as virtual machine sizes and number of nodes.

Technology Preview: MachinePool scaling with the console is in preview status, while the **MachinePool** resource is fully supported.

- Using the **MachinePool** resource is a feature that is not supported for bare metal clusters.
- A **MachinePool** resource is a Kubernetes resource on the hub cluster that groups the **MachineSet** resources together on the managed cluster.
- The **MachinePool** resource uniformly configures a set of machine resources, including zone configurations, instance type, and root storage.
- With **MachinePool**, you can manually configure the desired number of nodes or configure auto-scaling of nodes on the managed cluster.

1.4.8.1.1. Autoscaling

Configuring autoscaling provides the flexibility of your cluster to scale as needed to lower your cost of resources by scaling down when traffic is low, and by scaling up to ensure that there are enough resources when there is a higher demand for resources.

1.4.8.1.1.1. Enabling autoscaling

- To enable autoscaling on your **MachinePool** resources using the console, complete the following steps:
  1. In the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.
  2. Click the name of your target cluster and select the *Machine pools* tab.
  3. From the machine pools page, select **Enable autoscale** from the *Options* menu for the target machine pool.
  4. Select the minimum and maximum number of machine set replicas. A machine set replica maps directly to a node on the cluster.
     The changes might take several minutes to reflect on the console after you click **Scale**. You can view the status of the scaling operation by clicking **View machines** if the notification of the *Machine pools* tab.
- To enable autoscaling on your **MachinePool** resources using the command line, complete the following steps:
  1. Enter the following command to view your list of machine pools:

     ```
     oc get machinepools -n <managed-cluster-namespace>
     ```

     Replace **managed-cluster-namespace** with the namespace of your target managed cluster.
  2. Enter the following command to edit the YAML file for the machine pool:

     ```
     oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
     ```

Replace **name-of-MachinePool-resource** with the name of your **MachinePool** resource.

Replace **namespace-of-managed-cluster** with the name of the namespace of your managed cluster.

3. Delete the **spec.replicas** field from the YAML file.
4. Add the **spec.autoscaling.minReplicas** setting and **spec.autoscaling.maxReplicas** fields to the resource YAML.
5. Add the minimum number of replicas to the **minReplicas** setting.
6. Add the maximum number of replicas into the **maxReplicas** setting.
7. Save the file to submit the changes.

Autoscaling is enabled for the machine pool.

1.4.8.1.1.2. Disabling autoscaling

You can disable autoscaling by using the console or the command line.

- To disable autoscaling by using the console, complete the following steps:
  1. In the navigation, select **Infrastructure** > **Clusters**.
  2. Click the name of your target cluster and select the *Machine pools* tab.
  3. From the machine pools page, select **Disable autoscale** from the *Options* menu for the target machine pool.
  4. Select the number of machine set replicas that you want. A machine set replica maps directly with a node on the cluster.
     It might take several minutes to display in the console after you click **Scale**. You can view the status of the scaling by clicking **View machines** in the notification on the *Machine pools* tab.
- To disable autoscaling by using the command line, complete the following steps:
  1. Enter the following command to view your list of machine pools:

     ```
     oc get machinepools -n <managed-cluster-namespace>
     ```

     Replace **managed-cluster-namespace** with the namespace of your target managed cluster.
  2. Enter the following command to edit the YAML file for the machine pool:

     ```
     oc edit machinepool <name-of-MachinePool-resource> -n <namespace-of-managed-cluster>
     ```

     Replace **name-of-MachinePool-resource** with the name of your **MachinePool** resource.

     Replace **namespace-of-managed-cluster** with the name of the namespace of your managed cluster.
  3. Delete the **spec.autoscaling** field from the YAML file.

4. Add the **spec.replicas** field to the resource YAML.
5. Add the number of replicas to the **replicas** setting.
6. Save the file to submit the changes.

1.4.8.2. Scaling hosts to an infrastructure environment

You can use the console to add hosts to an infrastructure environment. Adding the hosts makes it easier to select the already-configured hosts when you are creating a cluster.

Complete the following steps to add a host:

1. From the navigation, select **Infrastructure** > **Host inventory**.
2. Select the host inventory where you want to add the host to view settings.
3. Select the *Hosts* tab to view the hosts that are already added to that environment and to add a host. Available hosts might take a few minutes to appear in the table.
4. Select **Using a Discovery ISO**, **With BMC form**, or **By uploading a YAML** to enter the information for your host.
5. If you select the **Using a Discovery ISO** option, complete the following steps:
   a. Copy the command that is provided in the console to download the ISO or select **Download Discovery ISO**.
   b. Run the command on a bootable device to start each host.
   c. For added security, select **Approve host** for each of the discovered hosts. This additional step offers some protection in case your ISO file is changed and run by an unauthorized person.
   d. Rename the hosts that are named, **localhost** to unique names.
6. If you select the **With BMC form** option, complete the following steps:
   Note: The BMC option for adding hosts can only be used when the platform of your hub cluster is bare metal, Red Hat OpenStack Platform, VMware vSphere, or was installed using the user-provisioned infrastructure (UPI) method and the platform is **None**.
   a. Add the connection details for the BMC of your host.
   b. Select **Add host** to start the boot process. The host is automatically booted by using the discovery ISO image, and is added to the list of hosts when it is started.
   When you add a host by using the BMC option, the host is automatically approved.
7. If you select the **By uploading a YAML** option, complete the following steps:
   Note: You can upload multiple hosts at the same time when you use the YAML option.
   a. Select **Download template** to fill out the provided YAML.
   b. Select **Upload** and navigate to the completed YAML template to add multiple hosts.

You can now create an on-premises cluster on this infrastructure environment. See Creating a cluster in an on-premises environment for more information about creating a cluster.

1.4.8.2.1. Adding nodes to existing OpenShift Container Platform clusters deployed with CIM

To add more nodes to an existing OpenShift Container Platform cluster deployed with CIM, complete the previous steps, then associate the new **Agent** custom resource definitions with your **ClusterDeployment** custom resource by adding them to the **agentSelector** parameter.

1.4.9. Using cluster proxy add-ons

In some environments, a managed cluster is behind a firewall and cannot be accessed directly by the hub cluster. To gain access, you can set up a proxy add-on to access the **kube-apiserver** of the managed cluster to provide a more secure connection.

Required access: Editor

To configure a cluster proxy add-on for a hub cluster and a managed cluster, complete the following steps:

1. Configure the **kubeconfig** file to access the managed cluster **kube-apiserver** by completing the following steps:

   a. Provide a valid access token for the managed cluster.
   Note: : You can use the corresponding token of the service account. You can also use the default service account that is in the default namespace.

      i. Export the **kubeconfig** file of the managed cluster by running the following command:

      ```
      export KUBECONFIG=<managed-cluster-kubeconfig>
      ```

      ii. Add a role to your service account that allows it to access pods by running the following commands:

      ```
      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. Run the following command to locate the secret of the service account token:

      ```
      oc get secret -n default | grep <default-token>
      ```

      Replace **default-token** with the name of your secret.

      iv. Run the following command to copy the token:

      ```
      export MANAGED_CLUSTER_TOKEN=$(kubectl -n default get secret <default-token> -o jsonpath={.data.token} | base64 -d)
      ```

      Replace **default-token** with the name of your secret.

   b. Configure the **kubeconfig** file on the Red Hat Advanced Cluster Management hub cluster.

      i. Export the current **kubeconfig** file on the hub cluster by running the following command:

      ```
      oc config view --minify --raw=true > cluster-proxy.kubeconfig
      ```

ii. Modify the **server** file with your editor. This example uses commands when using **sed**. Run **alias sed=gsed**, if you are using OSX.

```
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
```

Replace **cluster1** with a managed cluster name that you want to access.

iii. Delete the original user credentials by entering the following commands:

```
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. Add the token of the service account:

```
sed -i'' -e '$a\    token: '"$MANAGED_CLUSTER_TOKEN"'' cluster-proxy.kubeconfig
```

2. List all of the pods on the target namespace of the target managed cluster by running the following command:

```
oc get pods --kubeconfig=cluster-proxy.kubeconfig -n <default>
```

Replace the **default** namespace with the namespace that you want to use.

3. Access other services on the managed cluster. This feature is available when the managed cluster is a Red Hat OpenShift Container Platform cluster. The service must use **service-serving-certificate** to generate server certificates:

- From the managed cluster, use the following service account token:

```
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)
```

4. From the hub cluster, convert the certificate authority to a file by running the following command:

```
oc get configmap kube-root-ca.crt -o=jsonpath='{.data.ca\.crt}' > hub-ca.crt
```

1.4.10. Configuring a specific cluster management role

When you install multicluster engine for Kubernetes operator, the default configuration provides the **cluster-admin** role on the hub cluster. This permission enables you to create, manage, and import

managed clusters on the hub cluster. In some situations, you might want to limit the access to certain managed clusters that are managed by the hub cluster, rather than providing access to all of the managed clusters on the hub cluster.

You can limit access to certain managed clusters by defining a cluster role and applying it to a user or group. Complete the following steps to configure and apply a role:

1. Define the cluster role by creating a YAML file with the following content:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: <clusterrole-name>
rules:
- apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - managedclusters
  resourceNames:
  - <managed-cluster-name>
  verbs:
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
- apiGroups:
  - cluster.open-cluster-management.io
  resources:
  - managedclusters
  verbs:
  - create
- apiGroups:
  - ""
  resources:
  - namespaces
  resourceNames:
  - <managed-cluster-name>
  verbs:
  - create
  - get
  - list
  - watch
  - update
  - delete
  - deletecollection
  - patch
- apiGroups:
  - register.open-cluster-management.io
  resources:
  - managedclusters/accept
  resourceNames:
```

```
  - <managed-cluster-name>
  verbs:
  - update
```

Replace ***clusterrole-name*** with the name of the cluster role that you are creating.

Replace ***managed-cluster-name*** with the name of the managed cluster that you want the user to have access to.

2. Apply the **clusterrole** definition by entering the following command:

```
oc apply -f <filename>
```

Replace ***filename*** with the name of the YAML file that you created in the previous step.

3. Enter the following command to bind the **clusterrole** to a specified user or group:

```
oc adm policy add-cluster-role-to-user <clusterrole-name> <username>
```

Replace ***clusterrole-name*** with the name of the cluster role that you applied in the previous step. Replace ***username*** with the username to which you want to bind the cluster role.

1.4.11. Managing cluster labels

Add a label to your cluster to select group resources. See Labels and Selectors for more information.

You can add new labels, remove existing labels, and edit existing labels for your clusters.

To manage your labels, navigate to **Infrastructure** > **Clusters** and find your cluster in the *Clusters* table. Use the **Options** menu for the cluster to select **Edit labels**.

- To add a new label, enter a label in the *Edit labels* dialog box. Your entry must be in the following format: **Key=Value**. If you are adding multiple labels, separate the labels by pressing **enter**, adding a comma, or adding a space between the labels.
  Labels are only saved after you click **Save**.

- To remove an existing label, click the **Remove** icon for the label that you want to remove in the list.

- To update an existing label, you can reassign its key to a new value by adding a new label using the same key with a different value. For example, you can change **Key=Value** by entering **Key=NewValue** to update the value of **Key**.

Tip: You can also edit a cluster label from the cluster details page. In the navigation menu, click **Infrastructure** > **Clusters**. From the Clusters page, access the details page for the cluster by clicking the name of the cluster. Select the **Edit** icon in the *Labels* section. The *Edit labels* dialog box is displayed.

1.4.12. Configuring Ansible Tower tasks to run on managed clusters

multicluster engine for Kubernetes operator is integrated with Ansible Tower automation so that you can create prehook and posthook AnsibleJob instances that occur before or after creating or upgrading your clusters. Configuring prehook and posthook jobs for cluster destroy, and cluster scale actions are not supported.

Required access: Cluster administrator

- Prerequisites
- Configuring an AnsibleJob template to run on a cluster by using the console
- Creating an AnsibleJob template
- Viewing the status of an Ansible job

1.4.12.1. Prerequisites

You must meet the following prerequisites to run Ansible templates on your clusters:

- OpenShift Container Platform 4.6 or later
- Install the Ansible Automation Platform Resource Operator to connect Ansible jobs to the lifecycle of Git subscriptions. For best results when using the AnsibleJob to launch Ansible Tower jobs, the Ansible Tower job template should be idempotent when it is run. You can find the Ansible Automation Platform Resource Operator in the OpenShift Container Platform *OperatorHub*.

For more information about installing and configuring Ansible Tower automation, see Setting up Ansible Tower tasks.

1.4.12.2. Configuring an *AnsibleJob* template to run on a cluster by using the console

You can specify the Ansible job template that you want to use for a cluster when you create the cluster, when you import the cluster, or after you create the cluster.

To specify the template when creating or importing a cluster, select the Ansible template that you want to apply to the cluster in the *Automation* step. If there are no Ansible templates, click **Add automation template** to create one.

To specify the template after creating a cluster, click **Update automation template** in the action menu of an existing cluster. You can also use the **Update automation template** option to update an existing automation template.

1.4.12.3. Creating an AnsibleJob template

To initiate an Ansible job with a cluster installation or upgrade, you must create an Ansible job template to specify when you want the jobs to run. They can be configured to run before or after the cluster installs or upgrades.

To specify the details about running the Ansible template while creating a template, complete the steps in the console:

1. Select **Infrastructure** > **Automation** from the navigation.
2. Select the applicable path for your situation:
   - If you want to create a new template, click **Create Ansible template** and continue with step 3.
   - If you want to modify an existing template, click **Edit template** from the *Options* menu of the template that you want to modify and continue with step 5.

3. Enter a unique name for your template, which contains lowercase alphanumeric characters or a hyphen (-).

4. Select the credential that you want to use for the new template. To link an Ansible credential to an Ansible template, complete the following steps:

    a. From the navigation, select **Automation**. Any template in the list of templates that is not linked to a credential contains a **Link to credential** icon that you can use to link the template to an existing credential. Only the credentials in the same namespace as the template are displayed.

    b. If there are no credentials that you can select, or if you do not want to use an existing credential, select **Edit template** from the *Options* menu for the template that you want to link.

    c. Click **Add credential** and complete the procedure in Creating a credential for Ansible Automation Platform if you have to create your credential.

    d. After you create your credential in the same namespace as the template, select the credential in the *Ansible Automation Platform credential* field when you edit the template.

5. If you want to initiate any Ansible jobs before the cluster is installed, select **Add an Ansible job template** in the *Pre-install Ansible job templates* section.

6. Select or enter the name of the prehook and posthook Ansible jobs to add to the installation or upgrade of the cluster.
Note: The *Ansible job template name* must match the name of the Ansible job on the Ansible Tower.

7. Drag the Ansible jobs to change the order, if necessary.

8. Repeat steps 5 - 7 for any Ansible job templates that you want to initiate after the cluster is installed in the *Post-install Ansible job templates* section, the *Pre-upgrade Ansible job templates* section, and the *Post-upgrade Ansible job templates* section.

Your Ansible template is configured to run on clusters that specify this template when the designated actions occur.

1.4.12.4. Viewing the status of an Ansible job

You can view the status of a running Ansible job to ensure that it started, and is running successfully. To view the current status of a running Ansible job, complete the following steps:

1. In the menu, select **Infrastructure** > **Clusters** to access the *Clusters* page.

2. Select the name of the cluster to view its details.

3. View the status of the last run of the Ansible job on the cluster information. The entry shows one of the following statuses:

    - When an install prehook or posthook job fails, the cluster status shows **Failed**.

    - When an upgrade prehook or posthook job fails, a warning is displayed in the *Distribution* field that the upgrade failed.
    Tip: You can retry an upgrade from the *Clusters* page if the cluster prehook or posthook failed.

1.4.13. Creating and managing ManagedClusterSets

A **ManagedClusterSet** is a group of managed clusters. With a managed cluster set, you can manage access to all of the managed clusters in the group together. You can also create a **ManagedClusterSetBinding** resource to bind a **ManagedClusterSet** resource to a namespace.

Each managed cluster must be a member of a managed cluster set. When you install the hub cluster, a default **ManagedClusterSet** resource is created called **default**. All managed clusters that are not specifically assigned to a managed cluster set are automatically assigned to the **default** managed cluster set. To ensure that the default managed cluster set is always available, you cannot delete or update the **default** managed cluster set.

Note: Cluster pools that are not specifically added to a managed cluster set are not added to the default **ManagedClusterSet** resource. After a managed cluster is claimed from the cluster pool, the managed cluster is added to the default **ManagedClusterSet**.

1.4.13.1. Global ManagedClusterSet

When you create a managed cluster, the following are automatically created to ease management:

- A **ManagedClusterSet** called **global**.
- The namespace called **open-cluster-management-global-set**.
- A **ManagedClusterSetBinding** called **global** to bind the **global ManagedClusterSet** to the **open-cluster-management-global-set** namespace.
  Important: You cannot delete, update, or edit the **global** managed cluster set. The **global** managed cluster set includes all managed clusters. See the following example:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSetBinding
metadata:
  name: global
  namespace: open-cluster-management-global-set
spec:
  clusterSet: global
```

Continue reading to learn more on updating managed cluster sets:

- Creating a ManagedClusterSet
- Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet
- Creating a ManagedClusterSetBinding resource
- Adding a cluster to a ManagedClusterSet
- Removing a cluster from a ManagedClusterSet

1.4.13.2. Creating a ManagedClusterSet

You can group managed clusters together in a managed cluster set to limit the user access on managed clusters.

Required access: Cluster administrator

A **ManagedClusterSet** is a cluster-scoped resource, so you must have cluster administration permissions for the cluster where you are creating the **ManagedClusterSet**. A managed cluster cannot be included in more than one **ManagedClusterSet**. You can create a managed cluster set from either the multicluster engine for Kubernetes operator console or from the command-line interface.

1.4.13.2.1. Creating a ManagedClusterSet by using the console

Complete the following steps to create a managed cluster set by using the console:

1. In the main console navigation, select **Infrastructure** > **Clusters** and ensure that the *Cluster sets* tab is selected.
2. Select **Create cluster set**, and enter the name of the cluster set.

1.4.13.2.2. Creating a ManagedClusterSet by using the command line

Add the following definition of the managed cluster set to your **yaml** file to create a managed cluster set by using the command line:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSet
metadata:
  name: <clusterset1>
```

Replace **clusterset1** with the name of your managed cluster set.

1.4.13.3. Assigning users or groups Role-Based Access Control permissions to your ManagedClusterSet

You can assign users or groups to your cluster set that are provided by the configured identity providers on the hub cluster.

Required access: Cluster administrator

The **ManagedClusterSet** API offers three levels of RBAC permissions:

- Cluster set **admin**
  - Full access permissions to all of the cluster and cluster pool resources that are assigned to the managed cluster set.
  - Permission to create clusters, import clusters, and create cluster pools. The permissions must be assigned to the managed cluster set when the managed cluster set is created.
- Cluster set **bind**
  - Permission to bind the cluster set to a namespace by creating a **ManagedClusterSetBinding**. The user or group must also have permission to create the **ManagedClusterSetBinding** in the target namespace.
  - Read only permissions to all of the cluster and cluster pool resources that are assigned to the managed cluster set.
  - No permission to create clusters, import clusters, or create cluster pools.
- Cluster set **view**

- Read only permissions to all of the cluster and cluster pool resources that are assigned to the managed cluster set.
- No permission to create clusters, import clusters, or create cluster pools.

Note: You cannot apply the Cluster set **admin** permission for the global cluster set.

Complete the following steps to assign users or groups to your managed cluster set from the console:

1. From the main navigation menu of the console, select **Infrastructure** > **Clusters**.
2. Select the *Cluster sets* tab.
3. Select your target cluster set.
4. Select the *Access management* tab.
5. Select **Add user or group**.
6. Search for, and select the user or group that you want to provide access.
7. Select the **Cluster set admin** or **Cluster set view** role to give to the selected user or user group. See Overview of roles for more information about the role permissions.
8. Select **Add** to submit the changes.

Your user or group is displayed in the table. It might take a few seconds for the permission assignments for all of the managed cluster set resources to be propagated to your user or group.

For more information about role-based actions, see Role-based access control.

See Using ManagedClusterSets with Placement for placement information.

1.4.13.3.1. Creating a ManagedClusterSetBinding resource

Create a **ManagedClusterSetBinding** resource to bind a **ManagedClusterSet** resource to a namespace. Applications and policies that are created in the same namespace can only access managed clusters that are included in the bound managed cluster set resource.

Access permissions to the namespace automatically apply to a managed cluster set that is bound to that namespace. If you have access permissions to access the namespace to which the managed cluster set is bound, you automatically have permissions to access any managed cluster set that is bound to that namespace. However, if you only have permissions to access the managed cluster set, you do not automatically have permissions to access other managed cluster sets on the namespace. If you do not see a managed cluster set, you might not have the required permissions to view it.

You can create a managed cluster set binding by using the console or the command line.

1.4.13.3.1.1. Creating a ManagedClusterSetBinding by using the console

Complete the following steps to create a ManagedClusterSetBinding by using the console:

1. Access the cluster page by selecting **Infrastructure** > **Clusters** in the main navigation and select the *Cluster sets* tab.
2. Select the name of the cluster set that you want to create a binding for to view the cluster set details.

3. Select **Actions** > **Edit namespace bindings**.

4. On the *Edit namespace bindings* page, select the namespace to which you want to bind the cluster set from the drop-down menu. The existing namespaces that have bindings to the cluster set are already selected.

1.4.13.3.1.2. Creating a ManagedClusterSetBinding by using the command line

To create a managed cluster set binding by using the command line, complete the following steps:

1. Create the **ManagedClusterSetBinding** resource in your **yaml** file. When you create a managed cluster set binding, the name of the managed cluster set binding must match the name of the managed cluster set to bind. Your **ManagedClusterSetBinding** resource might resemble the following information:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: ManagedClusterSetBinding
metadata:
  namespace: project1
  name: clusterset1
spec:
  clusterSet: clusterset1
```

2. Ensure that you have the bind permission on the target managed cluster set. View the following example of a **ClusterRole** resource, which contains rules that allow the user to bind to **clusterset1**:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: clusterrole1
rules:
  - apiGroups: ["cluster.open-cluster-management.io"]
    resources: ["managedclustersets/bind"]
    resourceNames: ["clusterset1"]
    verbs: ["create"]
```

1.4.13.4. Adding a cluster to a ManagedClusterSet

After you create your **ManagedClusterSet**, you must add one or more managed clusters. You can add managed clusters to your managed cluster set by using either the console or the command line.

1.4.13.4.1. Adding clusters to a ManagedClusterSet by using the console

Complete the following steps to add a cluster to a managed cluster set by using the console:

1. If you just created your managed cluster set, select **Manage resource assignments** to go directly to the *Manage resource assignments* page. Continue with step 6 of this procedure.

2. If your cluster already exists, access the cluster page by selecting **Infrastructure** > **Clusters** in the main navigation.

3. Select the **Cluster sets** tab to view the available cluster sets.

4. Select the name of the cluster set that you want to add to the managed cluster sets to view the cluster set details.

5. Select **Actions** > **Manage resource assignments**.

6. On the *Manage resource assignments* page, select the checkbox for the resources that you want to add to the cluster set.

7. Select **Review** to review your changes.

8. Select **Save** to save your changes.
   Note: If you move a managed cluster from one managed cluster set to another, you must have the required RBAC permission available on both managed cluster sets.

1.4.13.4.2. Adding clusters to a ManagedClusterSet by using the command line

Complete the following steps to add a cluster to a managed cluster set by using the command line:

1. Ensure that there is an RBAC **ClusterRole** entry that allows you to create on a virtual subresource of **managedclustersets/join**. Without this permission, you cannot assign a managed cluster to a **ManagedClusterSet**.
   If this entry does not exist, add it to your **yaml** file. A sample entry resembles the following content:

   ```
   kind: ClusterRole
   apiVersion: rbac.authorization.k8s.io/v1
   metadata:
     name: clusterrole1
   rules:
     - apiGroups: ["cluster.open-cluster-management.io"]
       resources: ["managedclustersets/join"]
       resourceNames: ["<clusterset1>"]
       verbs: ["create"]
   ```

   Replace **clusterset1** with the name of your **ManagedClusterSet**.

   Note: If you are moving a managed cluster from one **ManagedClusterSet** to another, you must have that permission available on both managed cluster sets.

2. Find the definition of the managed cluster in the **yaml** file. The section of the managed cluster definition where you add a label resembles the following content:

   ```
   apiVersion: cluster.open-cluster-management.io/v1
   kind: ManagedCluster
   metadata:
     name: cluster1
   spec:
     hubAcceptsClient: true
   ```

   In this example, **cluster1** is the name of the managed cluster.

3. Add a label that specifies the name of the **ManagedClusterSet** in the format: **cluster.open-cluster-management.io/clusterset: clusterset1**.
   Your code resembles the following example:

   ```
   apiVersion: cluster.open-cluster-management.io/v1
   ```

```
kind: ManagedCluster
metadata:
  name: cluster1
  labels:
    cluster.open-cluster-management.io/clusterset: clusterset1
spec:
  hubAcceptsClient: true
```

In this example, **cluster1** is the cluster that is added to the managed cluster set names **clusterset1**.

Note: If the managed cluster was previously assigned to a managed cluster set that was deleted, the managed cluster might have a managed cluster set already specified to a cluster set that does not exist. If so, replace the name with the new one.

1.4.13.5. Using ManagedClusterSets with Placement

A **Placement** resource is a namespace-scoped resource that defines a rule to select a set of **ManagedClusters** from the **ManagedClusterSets**, which are bound to the placement namespace.

Required access: Cluster administrator, Cluster set administrator

1.4.13.5.1. Placement overview

See the following information about how placement with managed clusters works:

- Kubernetes clusters are registered with the hub cluster as cluster-scoped **ManagedClusters**.
- The **ManagedClusters** are organized into cluster-scoped **ManagedClusterSets**.
- The **ManagedClusterSets** are bound to workload namespaces.
- The namespace-scoped **Placements** specify a portion of **ManagedClusterSets** that select a working set of the potential **ManagedClusters**.
- **Placements** select from that working set by using label and claim selectors.
  Important: Placement does not select a **ManagedCluster** if there is no **ManagedClusterSet** bound to the placement namespace.
- The placement of **ManagedClusters** can be controlled by using taints and tolerations. See Using taints and tolerations to place managed clusters for more information.

The **Placement** specification includes the following fields:

- **ClusterSets** represents the **ManagedClusterSets** from which the **ManagedClusters** are selected.
  - If not specified, **ManagedClusters** is selected from the **ManagedClusterSets** that are bound to the placement namespace.
  - If specified, **ManagedClusters** is selected from the intersection of this set and the **ManagedClusterSets** that are bound to the placement namespace.
- **NumberOfClusters** represents the desired number of **ManagedClusters** to be selected, which meet the placement requirements.
  If not specified, all **ManagedClusters** that meet the placement requirements are selected.

- **Predicates** represents a slice of predicates to select **ManagedClusters** with label and claim selector. The predicates are ORed.
- **prioritizerPolicy** represents the policy of prioritizers.
  - **mode** is either **Exact**, **Additive**, or **""** where **""** is **Additive** by default.
    - In **Additive** mode, any prioritizer that is not specifically provided with configuration values is enabled with its default configurations. In the current default configuration, the *Steady* and *Balance* prioritizers have the weight of 1, while the other prioritizers have the weight of 0. The default configurations might change in the future, which might change the prioritization. **Additive** mode does not require that you configure all of the prioritizers.
    - In **Exact** mode, any prioritizer that is not specifically provided with configuration values has the weight of zero. **Exact** mode requires that you input the full set of prioritizers that you want, but avoids behavior changes between releases.
  - **configurations** represents the configuration of prioritizers.
    - scoreCoordinate represents the configuration of the prioritizer and score source.
      - **type** defines the type of the prioritizer score. Type is either **BuiltIn**, **AddOn**, or **" "**, where **" "** is **BuiltIn** by default. When the type is **BuiltIn**, the built in prioritizer name must be specified. When the type is **AddOn**, you need to configure the score source in **AddOn**.
      - **builtIn** defines the name of a BuiltIn prioritizer. The following list includes the valid **BuiltIn** prioritizer names:
        - Balance: Balance the decisions among the clusters.
        - Steady: Ensure the existing decision is stabilized.
        - ResourceAllocatableCPU & ResourceAllocatableMemory: Sort clusters based on the allocatable resources.
      - **addOn** defines the resource name and score name. **AddOnPlacementScore** is introduced to describe addon scores. See Extensible scheduling to learn more about it.
        - **resourceName** defines the resource name of the **AddOnPlacementScore**. The placement prioritizer selects the **AddOnPlacementScore** custom resource by this name.
        - **scoreName** defines the score name inside of the **AddOnPlacementScore**. **AddOnPlacementScore** contains a list of the score name and the score value. The **scoreName**, specifies the score to be used by the prioritizer.
    - **weight** defines the weight of prioritizer. The value must be in the range of [-10,10]. Each prioritizer calculates an integer score of a cluster in the range of [-100, 100]. The final score of a cluster is determined by the following formula **sum(weight * prioritizer_score)**. A higher weight indicates that the prioritizer receives a higher weight in the cluster selection, while 0 weight indicates that the prioritizer is disabled. A negative weight indicates that it is one of the last ones selected.

Note: The **configurations.name** file will be removed in v1beta1 and replaced by the **scoreCoordinate.builtIn** file. If both **name** and **scoreCoordinate.builtIn** are defined, the value in **scoreCoordinate.builtIn** is used to determine the selection.

1.4.13.5.2. Placement examples

You need to bind at least one **ManagedClusterSet** to a namespace by creating a **ManagedClusterSetBinding** in that namespace. **Note:** You need role-based access to **CREATE** on the virtual sub-resource of **managedclustersets/bind**. See the following examples:

- You can select **ManagedClusters** with the **labelSelector**. See the following sample where the **labelSelector** only matches clusters with label **vendor: OpenShift**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
```

- You can select **ManagedClusters** with **claimSelector**. See the following sample where **claimSelector** only matches clusters with **region.open-cluster-management.io** with **us-west-1**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement2
  namespace: ns1
spec:
  predicates:
    - requiredClusterSelector:
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

- You can select **ManagedClusters** from particular **clusterSets**. See the following sample where **claimSelector** only matches **clusterSets: clusterset1 clusterset2**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement3
  namespace: ns1
spec:
  clusterSets:
    - clusterset1
```

```
    - clusterset2
  predicates:
    - requiredClusterSelector:
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

- Select desired number of **ManagedClusters**. See the following sample where **numberOfClusters** is **3**:

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement4
  namespace: ns1
spec:
  numberOfClusters: 3
  predicates:
    - requiredClusterSelector:
        labelSelector:
          matchLabels:
            vendor: OpenShift
        claimSelector:
          matchExpressions:
            - key: region.open-cluster-management.io
              operator: In
              values:
                - us-west-1
```

- Select a cluster with the largest allocatable memory.
Note: Similar to Kubernetes Node Allocatable, 'allocatable' is defined as the amount of compute resources that are available for pods on each cluster.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement6
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
```

- Select a cluster with the largest allocatable CPU and memory, and make placement sensitive to resource changes.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement7
```

```
  namespace: ns1
spec:
  numberOfClusters: 1
  prioritizerPolicy:
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableCPU
        weight: 2
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
        weight: 2
```

- Select two clusters with the largest allocatable memory and the largest add-on score cpu ratio, and pin the placement decisions.

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement8
  namespace: ns1
spec:
  numberOfClusters: 2
  prioritizerPolicy:
    mode: Exact
    configurations:
      - scoreCoordinate:
          builtIn: ResourceAllocatableMemory
      - scoreCoordinate:
          builtIn: Steady
        weight: 3
      - scoreCoordinate:
          type: AddOn
          addOn:
            resourceName: default
            scoreName: cpuratio
```

1.4.13.5.3. Placement decision

One or multiple **PlacementDecisions** with label **cluster.open-cluster-management.io/placement={placement name}** are created to represent the **ManagedClusters** selected by a **Placement**.

If a **ManagedCluster** is selected and added to a **PlacementDecision**, components that consume this **Placement** might apply the workload on this **ManagedCluster**. After the **ManagedCluster** is no longer selected and it is removed from the **PlacementDecisions**, the workload that is applied on this **ManagedCluster** should be removed accordingly.

See the following **PlacementDecision** sample:

```
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: ''
```

1.4.13.5.4. Add-on status

You might want to select managed clusters for your placements according to the status of the add-ons that are deployed on them. For example, you want to select a managed cluster for your placement only if there is a specific add-on that is enabled on the cluster.

You can do this by specifying the label for the add-on, as well as its status, if necessary, when you create the Placement. A label is automatically created on a **ManagedCluster** resource if an add-on is enabled on the cluster. The label is automatically removed if the add-on is disabled.

Each add-on is represented by a label in the format of **feature.open-cluster-management.io/addon-<addon_name>=<status_of_addon>**.

Replace **addon_name** with the name of the add-on that should be enabled on the managed cluster that you want to select.

Replace **status_of_addon** with the status that the add-on should have if the cluster is selected. The possible values of **status_of_addon** are in the following list:

- **available**: The add-on is enabled and available.
- **unhealthy**: The add-on is enabled, but the lease is not updated continuously.
- **unreachable**: The add-on is enabled, but there is no lease found for it. This can also be caused when the managed cluster is offline.

For example, an available **application-manager** add-on is represented by a label on the managed cluster that reads:

```
feature.open-cluster-management.io/addon-application-manager: available
```

See the following examples of creating placements based on add-ons and their status:

- You can create a placement that includes all managed clusters that have **application-manager** enabled on them by adding the following YAML content:

  ```
  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
```

- You can create a placement that includes all managed clusters that have **application-manager** enabled with an **available** status by adding the following YAML content:

```
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"
```

- You can create a placement that includes all managed clusters that have **application-manager** disabled by adding the following YAML content:

```
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
```

1.4.13.5.5. Extensible scheduling

In placement resource-based scheduling, sometimes the prioritizer needs more data than the default value provided by the **MananagedCluster** resource to calculate the score of the managed cluster. For example, schedule the clusters based on CPU or memory usage data of the clusters that are fetched obtained through a monitoring system.

The API **AddOnPlacementScore** supports a more extensible way to schedule based on customized scores.

- You can specify the score in the **placement.yaml** file to select clusters.
- As a score provider, a 3rd party controller can run on either the hub cluster or the managed cluster, to maintain the lifecycle of **AddOnPlacementScore** and update score into it.

Refer to placement extensible scheduling enhancement in the **open-cluster-management** repository to learn more.

1.4.13.6. Using taints and tolerations to place managed clusters

You can control the placement of your managed clusters or managed cluster sets by using taints and tolerations. Taints and tolerations provide a way to prevent managed clusters from being selected for certain placements. This control can be helpful if you want to prevent certain managed clusters from being included in some placements. You can add a taint to the managed cluster, and add a toleration to the placement. If the taint and the toleration do not match, then the managed cluster is not selected for that placement.

1.4.13.6.1. Adding a taint to a managed cluster

Taints are specified in the properties of a managed cluster and allow a placement to repel a managed cluster or a set of managed clusters. You can add a taint to a managed cluster by entering a command that resembles the following example:

```
kubectl taint ManagedCluster <managed_cluster_name> key=value:NoSelect
```

The specification of a taint includes the following fields:

- **Required** Key - The taint key that is applied to a cluster. This value must match the value in the toleration for the managed cluster to meet the criteria for being added to that placement. You can determine this value. For example, this value could be **bar** or **foo.example.com/bar**.
- **Optional** Value - The taint value for the taint key. This value must match the value in the toleration for the managed cluster to meet the criteria for being added to that placement. For example, this value could be **value**.
- **Required** Effect - The effect of the taint on placements that do not tolerate the taint, or what occurs when the taint and the toleration of the placement do not match. The value of the effects must be one of the following values:
  - **NoSelect** - Placements are not allowed to select a cluster unless they tolerate this taint. If the cluster was selected by the placement before the taint was set, the cluster is removed from the placement decision.
  - **NoSelectIfNew** - The scheduler cannot select the cluster if it is a new cluster. Placements can only select the cluster if they tolerate the taint and already have the cluster in their cluster decisions.
- **Required TimeAdded** - The time when the taint was added. This value is automatically set.

1.4.13.6.2. Identifying built-in taints to reflect the status of managed clusters

When a managed cluster is not accessible, you do not want the cluster added to a placement. The following taints are automatically added to managed clusters that are not accessible:

- **cluster.open-cluster-management.io/unavailable** - This taint is added to a managed cluster when the cluster has a condition of **ManagedClusterConditionAvailable** with status of **False**. The taint has the effect of **NoSelect** and an empty value to prevent an unavailable cluster from being scheduled. An example of this taint is provided in the following content:

```
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** - This taint is added to a managed cluster when the status of the condition for **ManagedClusterConditionAvailable** is either **Unknown** or has no condition. The taint has effect of **NoSelect** and an empty value to prevent an unreachable cluster from being scheduled. An example of this taint is provided in the following content:

```
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'
```

1.4.13.6.3. Adding a toleration to a placement

Tolerations are applied to placements, and allow the placements to repel managed clusters that do not have taints that match the tolerations of the placement. The specification of a toleration includes the following fields:

- **Optional** Key - The key matches the taint key to allow the placement.
- **Optional** Value - The value in the toleration must match the value of the taint for the toleration to allow the placement.
- **Optional** Operator - The operator represents the relationship between a key and a value. Valid operators are **equal** and **exists**. The default value is **equal**. A toleration matches a taint when the keys are the same, the effects are the same, and the operator is one of the following values:
  - **equal** - The operator is **equal** and the values are the same in the taint and the toleration.
  - **exists** - The wildcard for value, so a placement can tolerate all taints of a particular category.
- **Optional** Effect - The taint effect to match. When left empty, it matches all taint effects. The allowed values when specified are **NoSelect** or **NoSelectIfNew**.
- **Optional** TolerationSeconds - The length of time, in seconds, that the toleration tolerates the taint before moving the managed cluster to a new placement. If the effect value is not **NoSelect** or **PreferNoSelect**, this field is ignored. The default value is **nil**, which indicates that there is no time limit. The starting time of the counting of the **TolerationSeconds** is automatically listed as the **TimeAdded** value in the taint, rather than in the value of the cluster scheduled time or the **TolerationSeconds** added time.

The following example shows how to configure a toleration that tolerates clusters that have taints:

- Taint on the managed cluster for this example:

```
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'
```

- Toleration on the placement that allows the taint to be tolerated

```
apiVersion: cluster.open-cluster-management.io/v1beta1
kind: Placement
metadata:
  name: placement1
  namespace: default
spec:
  tolerations:
    - key: gpu
      value: "true"
      operator: Equal
```

With the example tolerations defined, **cluster1** could be selected by the placement because the **key: gpu** and **value: "true"** match.

Note: A managed cluster is not guaranteed to be placed on a placement that contains a toleration for the taint. If other placements contain the same toleration, the managed cluster might be placed on one of those placements.

1.4.13.6.4. Specifying a temporary toleration

The value of **TolerationSeconds** specifies the period of time that the toleration tolerates the taint. This temporary toleration can be helpful when a managed cluster is offline and you can transfer applications that are deployed on this cluster to another managed cluster for a tolerated time.

For example, the managed cluster with the following taint becomes unreachable:

```
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'
```

If you define a placement with a value for **TolerationSeconds**, as in the following example, the workload transfers to another available managed cluster after 5 minutes.

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: Placement
metadata:
  name: demo4
  namespace: demo1
spec:
  tolerations:
    - key: cluster.open-cluster-management.io/unreachable
      operator: Exists
      tolerationSeconds: 300
----
```

The application is moved to another managed cluster after the managed cluster is unreachable for 5 minutes.

1.4.13.7. Removing a managed cluster from a ManagedClusterSet

You might want to remove a managed cluster from a managed cluster set to move it to a different managed cluster set, or remove it from the management settings of the set. You can remove a managed cluster from a managed cluster set by using the console or the command-line interface.

Note: Every managed cluster must be assigned to a managed cluster set. If you remove a managed cluster from a **ManagedClusterSet** and do not assign it to a different **ManagedClusterSet**, it is automatically added to the **default** managed cluster set.

1.4.13.7.1. Removing a managed cluster from a ManagedClusterSet by using the console

Complete the following steps to remove a cluster from a managed cluster set by using the console:

1. Click **Infrastructure** > **Clusters** and ensure that the *Cluster sets* tab is selected.
2. Select the name of the cluster set that you want to remove from the managed cluster set to view the cluster set details.
3. Select **Actions** > **Manage resource assignments**.
4. On the *Manage resource assignments* page, remove the checkbox for the resources that you want to remove from the cluster set.
   This step removes a resource that is already a member of the cluster set. You can see if the resource is already a member of a cluster set by viewing the details of the managed cluster.

Note: If you are moving a managed cluster from one managed cluster set to another, you must have the required RBAC permissions on both managed cluster sets.

1.4.13.7.2. Removing clusters from a ManagedClusterSet by using the command line

To remove a managed cluster from a managed cluster set by using the command line, complete the following steps:

1. Run the following command to display a list of managed clusters in the managed cluster set:

   ```
   oc get managedclusters -l cluster.open-cluster-management.io/clusterset=<clusterset1>
   ```

Replace **clusterset1** with the name of the managed cluster set.

2. Locate the entry for the cluster that you want to remove.

3. Remove the label from the the **yaml** entry for the cluster that you want to remove. See the following code for an example of the label:

```
labels:
  cluster.open-cluster-management.io/clusterset: clusterset1
```

Note: If you are moving a managed cluster from one managed cluster set to another, you must have the required RBAC permission available on both managed cluster sets.

1.4.14. Managing cluster pools (Technology Preview)

Cluster pools provide rapid and cost-effective access to configured Red Hat OpenShift Container Platform clusters on-demand and at scale. Cluster pools provision a configurable and scalable number of OpenShift Container Platform clusters on Amazon Web Services, Google Cloud Platform, or Microsoft Azure that can be claimed when they are needed. They are especially useful when providing or replacing cluster environments for development, continuous integration, and production scenarios. You can specify a number of clusters to keep running so that they are available to be claimed immediately, while the remainder of the clusters will be kept in a hibernating state so that they can be resumed and claimed within a few minutes.

ClusterClaim resources are used to check out clusters from cluster pools. When a cluster claim is created, the pool assigns a running cluster to it. If no running clusters are available, a hibernating cluster is resumed to provide the cluster or a new cluster is provisioned. The cluster pool automatically creates new clusters and resumes hibernating clusters to maintain the specified size and number of available running clusters in the pool.

- Creating a cluster pool
- Claiming clusters from cluster pools
- Updating the cluster pool release image
- Scaling cluster pools (Technology Preview)
- Destroying a cluster pool

The procedure for creating a cluster pool is similar to the procedure for creating a cluster. Clusters in a cluster pool are not created for immediate use.

1.4.14.1. Creating a cluster pool

The procedure for creating a cluster pool is similar to the procedure for creating a cluster. Clusters in a cluster pool are not created for immediate use.

Required access: Administrator

1.4.14.1.1. Prerequisites

See the following prerequisites before creating a cluster pool:

- You need to deploy a multicluster engine for Kubernetes operator hub cluster.

- You need Internet access for your multicluster engine for Kubernetes operator hub cluster so that it can create the Kubernetes cluster on the provider environment.
- You need an AWS, GCP, or Microsoft Azure provider credential. See Managing credentials overview for more information.
- You need a configured domain in your provider environment. See your provider documentation for instructions about how to configure a domain.
- You need provider login credentials.
- You need your OpenShift Container Platform image pull secret. See Using image pull secrets.

Note: Adding a cluster pool with this procedure configures it so it automatically imports the cluster to be managed by multicluster engine for Kubernetes operator when you claim a cluster from the pool. If you would like to create a cluster pool that does not automatically import the claimed cluster for management with the cluster claim, add the following annotation to your **clusterClaim** resource:

```
kind: ClusterClaim
metadata:
  annotations:
    cluster.open-cluster-management.io/createmanagedcluster: "false"
```

The word **"false"** must be surrounded by quotation marks to indicate that it is a string.

1.4.14.1.2. Create the cluster pool

To create a cluster pool, select **Infrastructure** > **Clusters** in the navigation menu. The *Cluster pools* tab lists the cluster pools that you can access. Select **Create cluster pool** and complete the steps in the console.

If you do not have a infrastructure credential that you want to use for the cluster pool, you can create one by selecting **Add credential**.

You can either select an existing namespace from the list, or type the name of a new one to create one. The cluster pool does not have to be in the same namespace as the clusters.

When you create a cluster pool in a cluster set, the **namespace admin** permission is applied to all of the users with **clusterset admin** permissions for the namespace where you add the cluster pool. Similarly, the **namespace view** permission is applied to the users with **clusterset view** permissions.

You can select a cluster set name if you want the RBAC roles for your cluster pool to share the role assignments of an existing cluster set. The cluster set for the clusters in the cluster pool can only be set when you create the cluster pool. You cannot change the cluster set association for the cluster pool or for the clusters in the cluster pool after you create the cluster pool. Any cluster that you claim from the cluster pool is automatically added to the same cluster set as the cluster pool.

Note: If you do not have **cluster admin** permissions, you must select a cluster set. The request to create a cluster set is rejected with a forbidden error if you do not include the cluster set name in this situation. If no cluster sets are available for you to select, contact your cluster administrator to create a cluster set and give you **clusterset admin** permissions to it.

The **cluster pool size** specifies the number of clusters that you want provisioned in your cluster pool, while the cluster pool running count specifies the number of clusters that the pool keeps running and ready to claim for immediate use.

The procedure is very similar to the procedure for creating clusters.

For specific information about the information that is required for your provider, see the following information:

- Creating a cluster on Amazon Web Services
- Creating a cluster on Google Cloud Platform
- Creating a cluster on Microsoft Azure

1.4.14.2. Claiming clusters from cluster pools

ClusterClaim resources are used to check out clusters from cluster pools. A claim is completed when a cluster is running and ready in the cluster pool. The cluster pool automatically creates new running and hibernated clusters in the cluster pool to maintain the requirements that are specified for the cluster pool.

Note: When a cluster that was claimed from the cluster pool is no longer needed and is destroyed, the resources are deleted. The cluster does not return to the cluster pool.

Required access: Administrator

1.4.14.2.1. Prerequisite

You must have the following available before claiming a cluster from a cluster pool:

A cluster pool with or without available clusters. If there are available clusters in the cluster pool, the available clusters are claimed. If there are no available clusters in the cluster pool, a cluster is created to fulfill the claim. See Creating a cluster pool for information about how to create a cluster pool.

1.4.14.2.2. Claim the cluster from the cluster pool

When you create a cluster claim, you request a new cluster from the cluster pool. A cluster is checked out from the pool when a cluster is available. The claimed cluster is automatically imported as one of your managed clusters, unless you disabled automatic import.

Complete the following steps to claim a cluster:

1. From the navigation menu, click **Infrastructure** > **Clusters**, and select the *Cluster pools* tab.
2. Find the name of the cluster pool you want to claim a cluster from and select **Claim cluster**.

If a cluster is available, it is claimed and immediately appears in the *Managed clusters* tab. If there are no available clusters, it might take several minutes to resume a hibernated cluster or provision a new cluster. During this time, the claim status is **pending**. Expand the cluster pool to view or delete pending claims against it.

The claimed cluster remains a member of the cluster set that it was associated with when it was in the cluster pool. You cannot change the cluster set of the claimed cluster when you claim it.

1.4.14.3. Updating the cluster pool release image

When the clusters in your cluster pool remain in hibernation for some time, the Red Hat OpenShift Container Platform release image of the clusters might become backlevel. If this happens, you can upgrade the version of the release image of the clusters that are in your cluster pool.

Required access: Edit

Complete the following steps to update the OpenShift Container Platform release image for the clusters in your cluster pool:

Note: This procedure does not update clusters from the cluster pool that are already claimed in the cluster pool. After you complete this procedure, the updates to the release images only apply to the following clusters that are related to the cluster pool:

- Clusters that are created by the cluster pool after updating the release image with this procedure.
- Clusters that are hibernating in the cluster pool. The existing hibernating clusters with the old release image are destroyed, and new clusters with the new release image replace them.

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. Find the name of the cluster pool that you want to update in the *Cluster pools* table.
4. Click the *Options* menu for the *Cluster pools* in the table, and select **Update release image**.
5. Select a new release image to use for future cluster creations from this cluster pool.

The cluster pool release image is updated.

Tip: You can update the release image for multiple cluster pools with one action by selecting the box for each of the the cluster pools and using the *Actions* menu to update the release image for the selected cluster pools.

1.4.14.4. Scaling cluster pools (Technology Preview)

You can change the number of clusters in the cluster pool by increasing or decreasing the number of clusters in the cluster pool size.

Required access: Cluster administrator

Complete the following steps to change the number of clusters in your cluster pool:

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. In the *Options* menu for the cluster pool that you want to change, select **Scale cluster pool**.
4. Change the value of the pool size.
5. Optionally, you can update the number of running clusters to increase or decrease the number of clusters that are immediately available when you claim them.

Your cluster pools are scaled to reflect your new values.

1.4.14.5. Destroying a cluster pool

If you created a cluster pool and determine that you no longer need it, you can destroy the cluster pool. When you destroy a cluster pool, all of the unclaimed hibernating clusters are destroyed and their resources are released.

Required access: Cluster administrator

To destroy a cluster pool, complete the following steps:

1. From the navigation menu, click **Infrastructure** > **Clusters**.
2. Select the *Cluster pools* tab.
3. In the *Options* menu for the cluster pool that you want to delete, select **Destroy cluster pool**. Any unclaimed clusters in the cluster pool are destroyed. It might take some time for all of the resources to be deleted, and the cluster pool remains visible in the console until all of the resources are deleted.
   The namespace that contained the ClusterPool will not be deleted. Deleting the namespace will destroy any clusters that have been claimed from the ClusterPool, since the ClusterClaim resources for these clusters are created in the same namespace.

Tip: You can destroy multiple cluster pools with one action by selecting the box for each of the the cluster pools and using the *Actions* menu to destroy the selected cluster pools.

1.4.15. ClusterClaims

A ***ClusterClaim*** is a cluster-scoped custom resource definition (CRD) on a managed cluster. A ClusterClaim represents a piece of information that a managed cluster claims. The following example shows a claim that is identified in the YAML file:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: id.openshift.io
spec:
  value: 95f91f25-d7a2-4fc3-9237-2ef633d8451c
```

The following table shows the defined ClusterClaims that might be on a cluster that multicluster engine for Kubernetes operator manages:

Claim name	Reserved	Mutable	Description
id.k8s.io	true	false	ClusterID defined in upstream proposal
kubeversion.open-cluster-management.io	true	true	Kubernetes version
platform.open-cluster-management.io	true	false	Platform the managed cluster is running on, like AWS, GCE, and Equinix Metal

Claim name	Reserved	Mutable	Description
product.open-cluster-management.io	true	false	Product name, like OpenShift, Anthos, EKS and GKE
id.openshift.io	false	false	OpenShift Container Platform external ID, which is only available for an OpenShift Container Platform cluster
consoleurl.openshift.io	false	true	URL of the management console, which is only available for an OpenShift Container Platform cluster
version.openshift.io	false	true	OpenShift Container Platform version, which is only available for an OpenShift Container Platform cluster

If any of the previous claims are deleted or updated on managed cluster, they are restored or rolled back to a previous version automatically.

After the managed cluster joins the hub, the ClusterClaims that are created on a managed cluster are synchronized with the status of the **ManagedCluster** resource on the hub. A managed cluster with ClusterClaims might look similar to the following example:

```
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: '4.5'
  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
```

1.4.15.1. List existing ClusterClaims

You can use the **kubectl** command to list the ClusterClaims that apply to your managed cluster. This is helpful when you want to compare your ClusterClaim to an error message.

Note: Make sure you have **list** permission on resource **clusterclaims.cluster.open-cluster-management.io**.

Run the following command to list all existing ClusterClaims that are on the managed cluster:

```
kubectl get clusterclaims.cluster.open-cluster-management.io
```

1.4.15.2. Create custom ClusterClaims

You can create ClusterClaims with custom names on a managed cluster, which makes it easier to identify them. The custom ClusterClaims are synchronized with the status of the **ManagedCluster** resource on the hub cluster. The following content shows an example of a definition for a customized **ClusterClaim**:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: ClusterClaim
metadata:
  name: <custom_claim_name>
spec:
  value: <custom_claim_value>
```

The max length of field **spec.value** is 1024. The **create** permission on resource **clusterclaims.cluster.open-cluster-management.io** is required to create a ClusterClaim.

1.4.16. Enabling ManagedServiceAccount add-ons (Technology Preview)

When you install the multicluster engine for Kubernetes operator, the **ManagedServiceAccount** add-on is disabled by default. This component when enabled allows you to create or delete a service account on a managed cluster.

Required access: Editor

When a **ManagedServiceAccount** custom resource is created in the **<managed_cluster>** namespace on the hub cluster, a **ServiceAccount** is created on the managed cluster.

A **TokenRequest** is made with the **ServiceAccount** on the managed cluster to the Kubernetes API server on the managed cluster. The token is then stored in a **Secret** in the **<target_managed_cluster>** namespace on the hub cluster.

Note: The token can expire and be rotated. See TokenRequest for more information about token requests.

1.4.16.1. Prerequisites

- Red Hat OpenShift Container Platform version 4.11 or later must be deployed in your environment, and you must be logged in with the command line interface (CLI).
- You need the multicluster engine for Kubernetes operator installed.

1.4.16.2. Enabling ManagedServiceAccount

To enable a **Managed-ServiceAccount** add-on for a hub cluster and a managed cluster, complete the following steps:

1. Enable the **ManagedServiceAccount** add-on on hub cluster. See Advanced configuration to learn more.
2. Deploy the **ManagedServiceAccount** add-on and apply it to your target managed cluster. Create the following YAML file and replace **target_managed_cluster** with the name of the managed cluster where you are applying the **Managed-ServiceAccount** add-on:

```
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. Run the following command to apply the file:

```
oc apply -f -
```

You have now enabled the **Managed-ServiceAccount** plugin for your managed cluster. See the following steps to configure a **ManagedServiceAccount**.

4. Create a **ManagedServiceAccount** custom resource with the following YAML source:

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

- Replace **managed_serviceaccount_name** with the name of your **ManagedServiceAccount**.
- Replace **target_managed_cluster** with the name of the managed cluster to which you are applying the **ManagedServiceAccount**.

5. To verify, view the **tokenSecretRef** attribute in the **ManagedServiceAccount** object status to find the secret name and namespace. Run the following command with your account and cluster name:

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

6. View the **Secret** containing the retrieved token that is connected to the created **ServiceAccount** on the managed cluster. Run the following command:

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

1.4.17. Upgrading your cluster

After you create Red Hat OpenShift Container Platform clusters that you want to manage with multicluster engine for Kubernetes operator, you can use the multicluster engine for Kubernetes operator console to upgrade those clusters to the latest minor version that is available in the version channel that the managed cluster uses.

In a connected environment, the updates are automatically identified with notifications provided for each cluster that requires an upgrade in the Red Hat Advanced Cluster Management console.

Notes:

To upgrade to a major version, you must verify that you meet all of the prerequisites for upgrading to that version. You must update the version channel on the managed cluster before you can upgrade the cluster with the console.

After you update the version channel on the managed cluster, the multicluster engine for Kubernetes operator console displays the latest versions that are available for the upgrade.

This method of upgrading only works for OpenShift Container Platform managed clusters that are in a *Ready* state.

Important: You cannot upgrade Red Hat OpenShift Kubernetes Service managed clusters or OpenShift Container Platform managed clusters on Red Hat OpenShift Dedicated by using the multicluster engine for Kubernetes operator console.

To upgrade your cluster in a connected environment, complete the following steps:

1. From the navigation menu, navigate to **Infrastructure** > **Clusters**. If an upgrade is available, it is shown in the *Distribution version* column.
2. Select the clusters in *Ready* state that you want to upgrade. A cluster must be an OpenShift Container Platform cluster to be upgraded with the console.
3. Select **Upgrade**.
4. Select the new version of each cluster.
5. Select **Upgrade**.

If your cluster upgrade fails, the Operator generally retries the upgrade a few times, stops, and reports the status of the failing component. In some cases, the upgrade process continues to cycle through attempts to complete the process. Rolling your cluster back to a previous version following a failed upgrade is not supported. Contact Red Hat support for assistance if your cluster upgrade fails.

1.4.17.1. Selecting a channel

You can use the Red Hat Advanced Cluster Management console to select a channel for your cluster upgrades on OpenShift Container Platform version 4.6, or later. After selecting a channel, you are automatically reminded of cluster upgrades that are available for both Errata versions (4.8.1 > 4.8.2 > 4.8.3, and so on) and release versions (4.8 > 4.9, and so on).

To select a channel for your cluster, complete the following steps:

1. From the Red Hat Advanced Cluster Management navigation, select **Infrastructure** > **Clusters**.
2. Select the name of the cluster that you want to change to view the *Cluster details* page. If a different channel is available for the cluster, an edit icon is displayed in the *Channel* field.
3. Click the edit icon to modify the setting in the field.
4. Select a channel in the *New channel* field.

You can find the reminders for the available channel updates in the *Cluster details* page of the cluster.

1.5. DISCOVERY SERVICE INTRODUCTION

You can discover OpenShift 4 clusters that are available from OpenShift Cluster Manager. After discovery, you can import your clusters to manage. The Discovery services uses the Discover Operator for back-end and console usage.

You must have an OpenShift Cluster Manager credential. See Creating a credential for Red Hat OpenShift Cluster Manager if you need to create a credential.

Required access: Administrator

- Configure Discovery with the console
- Configure Discovery using the CLI

1.5.1. Configure Discovery with the console

Use the product console to enable Discovery.

Required access: Access to the namespace where the credential was created.

1.5.1.1. Prerequisites

- You need a credential. See Creating a credential for Red Hat OpenShift Cluster Manager to connect to OpenShift Cluster Manager.

1.5.1.2. Configure Discovery

Configure Discovery in the console to find clusters. You can create multiple **DiscoveryConfig** resources with separate credentials. Follow instructions in the console.

1.5.1.3. View discovered clusters

After you set up your credentials and discover your clusters for import, you can view them in the console.

1. Click **Clusters** > **Discovered clusters**
2. View the populated table with the following information:
   - *Name* is the display name that is designated in OpenShift Cluster Manager. If the cluster does not have a display name, a generated name based on the cluster console URL is displayed. If the console URL is missing or was modified manually in OpenShift Cluster Manager, the cluster external ID is displayed.
   - *Namespace* is the namespace where you created the credential and discovered clusters.
   - *Type* is the discovered cluster Red Hat OpenShift type.
   - *Distribution version* is the discovered cluster Red Hat OpenShift version.
   - *Infrastructure provider* is the cloud provider of the discovered cluster.
   - *Last active* is the last time the discovered cluster was active.
   - *Created* when the discovered cluster was created.
   - *Discovered* when the discovered cluster was discovered.
3. You can search for any information in the table, as well. For example, to show only *Discovered clusters* in a particular namespace, search for that namespace.
4. You can now click **Import cluster** to create managed clusters. See Import discovered clusters.

1.5.1.4. Import discovered clusters

After you discover clusters, you can import clusters that appear in the *Discovered clusters* tab of the console.

1.5.1.5. Prerequisites

You need access to the namespaces that were used to configure Discovery.

1.5.1.6. Import Discovered clusters

1. Navigate to the existing *Clusters* page and click on the *Discovered clusters* tab.
2. From the *Discovered clusters* table, find the cluster that you want to import.
3. From the options menu, choose **Import cluster**.
4. For discovered clusters, you can import manually using the documentation, or you can choose Import clusters automatically.
5. To import automatically with your credentials or Kubeconfig file, copy and paste the content.
6. Click **Import**.

1.5.2. Enable Discovery using the CLI

Enable discovery using the CLI to find clusters that are available from Red Hat OpenShift Cluster Manager.

Required access: Administrator

1.5.2.1. Prerequisites

- Create a credential to connect to Red Hat OpenShift Cluster Manager.

1.5.2.2. Discovery set up and process

Note: The **DiscoveryConfig** must be named **discovery** and must be created in the same namespace as the selected **credential**. See the following **DiscoveryConfig** sample:

```
apiVersion: discovery.open-cluster-management.io/v1
kind: DiscoveryConfig
metadata:
  name: discovery
  namespace: <NAMESPACE_NAME>
spec:
  credential: <SECRET_NAME>
  filters:
    lastActive: 7
    openshiftVersions:
    - "4.10"
    - "4.11"
    - "4.8"
```

1. Replace **SECRET_NAME** with the credential that you previously set up.
2. Replace **NAMESPACE_NAME** with the namespace of **SECRET_NAME**.
3. Enter the maximum time since last activity of your clusters (in days) to discover. For example, with **lastActive: 7**, clusters that active in the last 7 days are discovered.

4. Enter the versions of Red Hat OpenShift clusters to discover as a list of strings. **Note:** Every entry in the **openshiftVersions** list specifies an OpenShift major and minor version. For example, specifying **"4.11"** will include all patch releases for the OpenShift version **4.11**, for example **4.11.1**, **4.11.2**.

1.5.2.3. View discovered clusters

View discovered clusters by running **oc get discoveredclusters -n <namespace>** where **namespace** is the namespace where the discovery credential exists.

1.5.2.3.1. DiscoveredClusters

Objects are created by the Discovery controller. These **DiscoveredClusters** represent the clusters that are found in OpenShift Cluster Manager by using the filters and credentials that are specified in the **DiscoveryConfig discoveredclusters.discovery.open-cluster-management.io** API. The value for **name** is the cluster external ID:

```
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: 4.10
  status: Stale
```

1.6. USING HOSTED CONTROL PLANE CLUSTERS (TECHNOLOGY PREVIEW)

multicluster engine for Kubernetes operator version 2.5 with the multicluster engine for Kubernetes operator 2.0 can deploy Red Hat OpenShift Container Platform clusters by using two different control plane configurations. The standalone configuration uses multiple dedicated virtual machines or physical machines to host the OpenShift Container Platform control plane. You can provision hosted control planes to provision the OpenShift Container Platform control plane as pods on a hosting service cluster without the need for dedicated physical machines for each control-plane.

Note: This feature also works with the multicluster engine for Kubernetes operator 2.0 without multicluster engine for Kubernetes operator.

Amazon Web Services is supported as a Technology Preview. You can host the control planes for your Red Hat OpenShift Container Platform version 4.10.7 and later.

The control plane is run as pods that are contained in a single namespace and is associated with the hosted control plane cluster. When OpenShift Container Platform provisions this type of hosted cluster, it provisions a worker node independent of the control plane.

See the following benefits of hosted control plane clusters:

- Saves cost by removing the need to host dedicated control plane nodes
- Introduces separation between the control plane and the workloads, which improves isolation and reduces configuration errors that can require changes
- Significantly decreases the cluster provisioning time by removing the requirement for control-plane node bootstrapping
- Supports turn-key deployments or fully customized OpenShift Container Platform provisioning

See more in the following product documentation:

- Configuring hosted control planes
- Disabling hosted control plane resources

1.6.1. Configuring hosted control planes

Configuring hosted control planes requires a hosting service cluster and a hosted cluster. By deploying the HyperShift operator on an existing cluster, you can make that cluster into a hosting service cluster and start the creation of the hosted cluster.

Hosted control planes is a Technology Preview feature, so the related components are disabled by default. Enable the feature by editing the **multiclusterengine** custom resource to set the **spec.overrides.components[?(@.name=='hypershift-preview')].enabled** to **true**.

Enter the following command to ensure that the hosted control planes feature is enabled:

```
oc patch mce multiclusterengine-sample --type=merge -p '{"spec":{"overrides":{"components":
[{"name":"hypershift-preview","enabled": true}]}}}'
```

1.6.1.1. Configuring the hosting service cluster

You can deploy hosted control planes by configuring an existing cluster to function as a hosting service cluster. The hosting service cluster is the OpenShift Container Platform cluster where the control planes are hosted, and can be the hub cluster or one of the OpenShift Container Platform managed clusters.

Best practice: Run hosted control planes and worker nodes on the same environment.

1.6.1.1.1. Prerequisites

You must have the following prerequisites to configure a hosting service cluster:

- multicluster engine for Kubernetes operator installed on at least one cluster that is managed by Red Hat OpenShift Container Platform. The multicluster engine for Kubernetes operator is automatically installed when you install Red Hat Advanced Cluster Management version 2.5, and later, and can also be installed without Red Hat Advanced Cluster Management as an operator from the OpenShift Container Platform OperatorHub.

- If you want your hub cluster to be your hosting service cluster, you must configure a **local-cluster** as your hosting service cluster by completing the steps in the *Local-cluster enablement* section of the Advanced configuration documentation.

1.6.1.1.2. Configure the hosting service cluster

Complete the following steps on the cluster where the multicluster engine for Kubernetes operator is installed to enable an OpenShift Container Platform managed cluster as a hosting service cluster:

1. If you plan to create and manage hosted clusters on AWS, create an OIDC S3 credentials secret named **hypershift-operator-oidc-provider-s3-credentials** for the HyperShift operator. Save the secret in the managed cluster namespace, which is the namespace of the managed cluster that is used as the hosting service cluster. If you used **local-cluster**, then create the secret in the **local-cluster** namespace
   The secret must contain 3 fields. The **bucket** field contains an S3 bucket with public access to host OIDC discovery documents for your HyperShift clusters. The **credentials** field is a reference to a file that contains the credentials of the **default** profile that can access the bucket. By default, HyperShift only uses the **default** profile to operate the **bucket**. The **region** field specifies the region of the S3 bucket.

   See Getting started in the HyperShift documentation for more information about the secret. The following example shows a sample AWS secret template:

   ```
   oc create secret generic hypershift-operator-oidc-provider-s3-credentials --from-file=credentials=$HOME/.aws/credentials --from-literal=bucket=<s3-bucket-for-hypershift> --from-literal=region=<region> -n <hypershift-hosting-service-cluster>
   ```

   Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add the label that enables the **hypershift-operator-oidc-provider-s3-credentials** secret to be backed up for disaster recovery:

   ```
   oc label secret hypershift-operator-oidc-provider-s3-credentials -n <hypershift-hosting-service-cluster> cluster.open-cluster-management.io/backup=""
   ```

2. If you plan to provision hosted clusters on the AWS platform with Private Link, create an AWS credential secret for the HyperShift operator and name it **hypershift-operator-private-link-credentials**. It must reside in the managed cluster namespace that is the namespace of the managed cluster being used as the hosting service cluster. If you used **local-cluster**, create the secret in the **local-cluster** namespace. See steps 1-5 in Deploy AWS private clusters for more details.
   The secret must contain the following three fields:

   - **aws-access-key-id**: AWS credential access key id
   - **aws-secret-access-key**: AWS credential access key secret
   - **region**: Region for use with Private Link
     See Getting started in the HyperShift documentation for more information about the secret. The following example shows the sample **hypershift-operator-private-link-credentials** secret template:

     ```
     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 <hypershift-hosting-service-cluster>
     ```

Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add the label that enables the **hypershift-operator-private-link-credentials** secret to be backed up for disaster recovery:

```
oc label secret hypershift-operator-private-link-credentials -n <hypershift-hosting-service-cluster> cluster.open-cluster-management.io/backup=""
```

Set the following parameter in the **HypershiftDeployment** custom resource when creating a cluster:

```
spec:
  hostedClusterSpec:
    platform:
      type: AWS
      aws:
        endpointAccess: Private
```

3. If you plan to use service-level DNS for control plane services on the AWS platform, create an external DNS credential secret for the HyperShift operator and name it **hypershift-operator-external-dns-credentials**. It must reside in the managed cluster namespace that is the namespace of the managed cluster being used as the hosting service cluster. If you used **local-cluster**, then create the secret in the **local-cluster** namespace
The secret must contain the following three fields:

   - **provider**: DNS provider that manages the service-level DNS zone (example: aws)
   - **domain-filter**: The service-level domain
   - **credentials**: (Optional, only when using aws keys) - For all external DNS types, a credential file is supported
   - **aws-access-key-id**: (Optional) - When using AWS DNS service, credential access key id
   - **aws-secret-access-key**: (Optional) - When using AWS DNS service, When using AWS DNS service, credential access key secret
   See Use Service-level DNS for Control Plane Services for more details. The following example shows the sample **hypershift-operator-external-dns-credentials** secret template:

```
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 <hypershift-hosting-service-cluster>
```

Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add the label that enables the **hypershift-operator-external-dns-credentials** secret to be backed up for disaster recovery:

```
oc label secret hypershift-operator-external-dns-credentials -n <hypershift-hosting-service-cluster> cluster.open-cluster-management.io/backup=""
```

Set the following parameter in the **HypershiftDeployment** custom resource when creating a cluster:

```
spec:
  hostedClusterSpec:
```

```
  platform:
    type: AWS
    aws:
      endpointAccess: PublicAndPrivate
```

4. Install the HyperShift add-on.
   The cluster that hosts the HyperShift operator is the hosting service cluster. This step uses the **hypershift-addon** to install the HyperShift operator on a managed cluster.

   a. Create a namespace where the HyperShift operator is created.

   b. Create the **ManagedClusterAddon** HyperShift add-on by creating a file that resembles the following example:

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

   Replace **managed-cluster-name** with the name of the managed cluster on which you want to install the HyperShift operator. If you are installing on the Red Hat Advanced Cluster Management hub cluster, then use **local-cluster** for this value.

   c. Apply the file by running the following command:

```
oc apply -f <filename>
```

   Replace **filename** with the name of the file that you created.

5. Confirm that the **hypershift-addon** is installed by running the following command:

```
oc get managedclusteraddons -n <hypershift-hosting-service-cluster> hypershift-addon
```

   The output resembles the following example, when the add-on is installed:

```
NAME              AVAILABLE   DEGRADED   PROGRESSING
hypershift-addon   True
```

Your HyperShift add-on is installed and the hosting service cluster is available to manage HyperShift clusters.

1.6.1.2. Deploying a hosted cluster

After installing the HyperShift operator and enabling an existing cluster as a hosting service cluster, you can provision a HyperShift hosted cluster by creating a **HypershiftDeployment** custom resource.

1. Create a cloud provider secret as a credential using the console or a file addition. You must have permissions to create infrastructure resources for your cluster, like VPCs, subnets, and NAT gateways. The account also must correspond to the account for your guest cluster, where your workers live. See Create AWS infrastructure and IAM resources separately in the HyperShift documentation for more information about the required permissions.

The following example shows the secret format for AWS:

```
apiVersion: v1
metadata:
  name: my-aws-cred
  namespace: default      # Where you create HypershiftDeployment resources
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
```

- To create this secret with the console, follow the credential creation steps by accessing **Credentials** in the navigation menu.
- To create the secret using the command line, run the following commands:

```
oc create secret generic <my-secret> -n <hypershift-deployment-namespace> --from-literal=baseDomain='your.domain.com' --from-literal=aws_access_key_id='your-aws-access-key' --from-literal=aws_secret_access_key='your-aws-secret-key' --from-literal=pullSecret='your-quay-pull-secret' --from-literal=ssh-publickey='your-ssh-publickey' --from-literal=ssh-privatekey='your-ssh-privatekey'
```

Note: Disaster recovery backup for the secret is not automatically enabled. Run the following command to add a label that enables the secret to be backed up for disaster recovery:

```
oc label secret <my-secret> -n <hypershift-deployment-namespace> cluster.open-cluster-management.io/backup=""
```

2. Create a **HypershiftDeployment** custom resource. The **HypershiftDeployment** custom resource creates the infrastructure in the provider account, configures the infrastructure compute capacity in the created infrastructure, provisions the **nodePools** that use the hosted control plane, and creates a hosted control plane on a hosting service cluster.

   a. Create a file that contains information that resembles the following example:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: HypershiftDeployment
metadata:
  name: <cluster>
  namespace: default
spec:
  hostingCluster: <hosting-service-cluster>
  hostingNamespace: clusters
  infrastructure:
    cloudProvider:
      name: <my-secret>
    configure: True
```

```
    platform:
      aws:
        region: <region>
```

Replace **cluster** with the name of the cluster.

Replace **hosting-service-cluster** with the name of the cluster that hosts the HyperShift operator.

Replace **my-secret** with the secret to access your cloud provider.

Replace **region** with the region of your cloud provider.

b. Apply the file by entering the following command:

```
oc apply -f <filename>
```

You can refer to the field definitions of the API to ensure that they are correct.

3. Check the **HypershiftDeployment** status by running the following command:

```
oc get hypershiftdeployment -n default hypershift-demo -w
```

4. After the hosted cluster is created, it is automatically imported to the hub. You can verify this by viewing the cluster list in the console, or by running the following command:

```
oc get managedcluster <hypershiftDeployment.Spec.infraID>
```

1.6.1.3. Deploying a customized hosted cluster

The **HypershiftDeployment** custom resource that you created provisions the hosted control plane and its nodePools with default values. You can also customize the hosted control plane and the nodePools by specifying **hostedClusterSpec** and **nodePools** in the **HypershiftDeployment** custom resource.

Create a file that contains information that resembles the following example:

```
apiVersion: cluster.open-cluster-management.io/v1alpha1
kind: HypershiftDeployment
metadata:
  name: <cluster>
  namespace: default
spec:
  hostingCluster: <hosting-service-cluster>
  hostingNamespace: clusters
  hostedClusterSpec:
    networking:
      machineCIDR: 10.0.0.0/16    # Default
      networkType: OVNKubernetes
      podCIDR: 10.132.0.0/14      # Default
      serviceCIDR: 172.31.0.0/16  # Default
    platform:
      type: AWS
    pullSecret:
      name: <cluster>-pull-secret    # This secret is created by the controller
    release:
```

```
    image: quay.io/openshift-release-dev/ocp-release:4.11.2-x86_64  # Default
  services:
  - service: APIServer
    servicePublishingStrategy:
      type: LoadBalancer
  - service: OAuthServer
    servicePublishingStrategy:
      type: Route
  - service: Konnectivity
    servicePublishingStrategy:
      type: Route
  - service: Ignition
    servicePublishingStrategy:
      type: Route
  sshKey: {}
nodePools:
- name: <cluster>-<zone-1>
  spec:
    clusterName: <cluster>
    management:
      autoRepair: false
      replace:
        rollingUpdate:
          maxSurge: 1
          maxUnavailable: 0
        strategy: RollingUpdate
      upgradeType: Replace
    platform:
      aws:
        instanceType: m5.large
      type: AWS
    release:
      image: quay.io/openshift-release-dev/ocp-release:4.10.15-x86_64
    replicas: 3
- name: <cluster>-<zone-2>
  spec:
    clusterName: <cluster>
    management:
      autoRepair: false
      replace:
        rollingUpdate:
          maxSurge: 1
          maxUnavailable: 0
        strategy: RollingUpdate
      upgradeType: Replace
    platform:
      aws:
        instanceType: m5.large
      type: AWS
    release:
      image: quay.io/openshift-release-dev/ocp-release:4.10.15-x86_64
    replicas: 3
infrastructure:
  cloudProvider:
    name: <my-secret>
  configure: True
```

```
platform:
  aws:
    region: <region>
    zones:
      - <zone-1>
      - <zone-2>
```

Replace **zone-1** with the name of the zone in the region.

Replace **zone-2** with the name of the zone in the region.

1.6.1.4. Accessing a hosting service cluster

You can now access your cluster. The access secrets are stored in the **hypershift-hosting-service-cluster** namespace. This namespace is the same as the name of the hosting service cluster. Learn about the following formats secret name formats:

- **kubeconfig** secret: **<hypershiftDeployment.Spec.hostingNamespace>-<hypershiftDeployment.Name>-admin-kubeconfig** (clusters-hypershift-demo-admin-kubeconfig)
- **kubeadmin** password secret: **<hypershiftDeployment.Spec.hostingNamespace>-<hypershiftDeployment.Name>-kubeadmin-password** (clusters-hypershift-demo-kubeadmin-password)

1.6.2. Disabling the hosted control plane resources

When disabling the hosted control plane cluster feature, you must destroy the HyperShift hosted cluster and uninstall the HyperShift operator.

1.6.2.1. Destroying a HyperShift hosted cluster

To destroy a HyperShift hosted cluster, delete the **HypershiftDeployment** resource by running one of the following commands:

```
oc delete -f <HypershiftDeployment_yaml_file_name>
```

or

```
oc delete hd -n <HypershiftDeployment_namespace> <HypershiftDeployment_resource_name>
```

1.6.2.2. Uninstalling the HyperShift operator

To uninstall the HyperShift operator from a management or hosting service cluster, delete the **hypershift-addon ManagedClusterAddon** from the management cluster by running the following command:

```
oc delete managedclusteraddon -n <hypershift-management-cluster> hypershift-addon
```

1.7. APIS

You can access APIs for the multicluster engine for Kubernetes operator for cluster lifecycle management. **User required access:** You can only perform actions that your role is assigned. For more information, review the API documentation for each of the following resources:

- Clusters API
- ClusterSets API (v1beta1)
- Clusterview API
- ClusterSetBindings API (v1beta1)
- MultiClusterEngine API
- Placements API (v1beta1)
- PlacementDecisions API (v1beta1)
- Managed service account (Technology Preview)

1.7.1. Clusters API

1.7.1.1. Overview

This documentation is for the cluster resource for multicluster engine for Kubernetes. Cluster resource has four possible requests: create, query, delete and update.

1.7.1.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.1.1.2. Tags

- cluster.open-cluster-management.io : Create and manage clusters

1.7.1.2. Paths

1.7.1.2.1. Query all clusters

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

1.7.1.2.1.1. Description

Query your clusters for more details.

1.7.1.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.1.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.1.2.1.4. Consumes

- **cluster/yaml**

1.7.1.2.1.5. Tags

- cluster.open-cluster-management.io

1.7.1.2.2. Create a cluster

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

1.7.1.2.2.1. Description

Create a cluster

1.7.1.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the cluster to be created.	Cluster

1.7.1.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content

HTTP Code	Description	Schema
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.1.2.2.4. Consumes

- **cluster/yaml**

1.7.1.2.2.5. Tags

- cluster.open-cluster-management.io

1.7.1.2.2.6. Example HTTP request

1.7.1.2.2.6.1. Request 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" : { }
}
```

1.7.1.2.3. Query a single cluster

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

1.7.1.2.3.1. Description

Query a single cluster for more details.

1.7.1.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**cluster_name** *required*	Name of the cluster that you want to query.	string

1.7.1.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.1.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.1.2.4. Delete a cluster

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

1.7.1.2.4.1. Description

Delete a single cluster

1.7.1.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**cluster_name** *required*	Name of the cluster that you want to delete.	string

1.7.1.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.1.2.4.4. Tags

- cluster.open-cluster-management.io

1.7.1.3. Definitions

1.7.1.3.1. Cluster

Name	Schema
apiVersion *required*	string
kind *required*	string
metadata *required*	object
spec *required*	spec

spec

Name	Schema
hubAcceptsClient *required*	bool
managedClusterClientConfigs *optional*	< managedClusterClientConfigs > array
leaseDurationSeconds *optional*	integer (int32)

managedClusterClientConfigs

Name	Description	Schema
URL *required*		string
CABundle *optional*	Pattern : **"^(?:[A-Za-z0-9+/]{4})*(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?$"**	string (byte)

1.7.2. Clustersets API (v1beta1)

1.7.2.1. Overview

This documentation is for the Clusterset resource for multicluster engine for Kubernetes. Clusterset resource has four possible requests: create, query, delete and update.

1.7.2.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.2.1.2. Tags

- cluster.open-cluster-management.io : Create and manage Clustersets

1.7.2.2. Paths

1.7.2.2.1. Query all clustersets

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

1.7.2.2.1.1. Description

Query your Clustersets for more details.

1.7.2.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.2.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.2.2.1.4. Consumes

- **clusterset/yaml**

1.7.2.2.1.5. Tags

- cluster.open-cluster-management.io

1.7.2.2.2. Create a clusterset

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

1.7.2.2.2.1. Description

Create a Clusterset.

1.7.2.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the clusterset to be created.	Clusterset

1.7.2.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.2.2.2.4. Consumes

- **clusterset/yaml**

1.7.2.2.2.5. Tags

- cluster.open-cluster-management.io

1.7.2.2.2.6. Example HTTP request

1.7.2.2.2.6.1. Request body

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

1.7.2.2.3. Query a single clusterset

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

1.7.2.2.3.1. Description

Query a single clusterset for more details.

1.7.2.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterset_name** *required*	Name of the clusterset that you want to query.	string

1.7.2.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.2.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.2.2.4. Delete a clusterset

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

1.7.2.2.4.1. Description

Delete a single clusterset.

1.7.2.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**clusterset_name** *required*	Name of the clusterset that you want to delete.	string

1.7.2.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.2.2.4.4. Tags

- cluster.open-cluster-management.io

1.7.2.3. Definitions

1.7.2.3.1. Clusterset

Name	Schema
apiVersion *required*	string
kind *required*	string
metadata *required*	object

1.7.3. Clustersetbindings API (v1beta1)

1.7.3.1. Overview

This documentation is for the clustersetbinding resource for multicluster engine for Kubernetes. Clustersetbinding resource has four possible requests: create, query, delete and update.

1.7.3.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.3.1.2. Tags

- cluster.open-cluster-management.io : Create and manage clustersetbindings

1.7.3.2. Paths

1.7.3.2.1. Query all clustersetbindings

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

1.7.3.2.1.1. Description

Query your clustersetbindings for more details.

1.7.3.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string

1.7.3.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.3.2.1.4. Consumes

- **clustersetbinding/yaml**

1.7.3.2.1.5. Tags

- cluster.open-cluster-management.io

1.7.3.2.2. Create a clustersetbinding

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

1.7.3.2.2.1. Description

Create a clustersetbinding.

1.7.3.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string
Body	**body** *required*	Parameters describing the clustersetbinding to be created.	Clustersetbinding

1.7.3.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.3.2.2.4. Consumes

- **clustersetbinding/yaml**

1.7.3.2.2.5. Tags

- cluster.open-cluster-management.io

1.7.3.2.2.6. Example HTTP request

1.7.3.2.2.6.1. Request body

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

1.7.3.2.3. Query a single clustersetbinding

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

1.7.3.2.3.1. Description

Query a single clustersetbinding for more details.

1.7.3.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string
Path	**clustersetbinding_name** *required*	Name of the clustersetbinding that you want to query.	string

1.7.3.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content

HTTP Code	Description	Schema
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.3.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.3.2.4. Delete a clustersetbinding

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

1.7.3.2.4.1. Description

Delete a single clustersetbinding.

1.7.3.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**namespace** *required*	Namespace that you want to use, for example, default.	string
Path	**clustersetbinding_name** *required*	Name of the clustersetbinding that you want to delete.	string

1.7.3.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content

HTTP Code	Description	Schema
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.3.2.4.4. Tags

- cluster.open-cluster-management.io

1.7.3.3. Definitions

1.7.3.3.1. Clustersetbinding

Name	Schema
apiVersion *required*	string
kind *required*	string
metadata *required*	object
spec *required*	spec

spec

Name	Schema
clusterSet *required*	string

1.7.4. Clusterview API (v1alpha1)

1.7.4.1. Overview

This documentation is for the **clusterview** resource for multicluster engine for Kubernetes. The **clusterview** resource provides a CLI command that enables you to view a list of the managed clusters and managed cluster sets that that you can access. The three possible requests are: list, get, and watch.

1.7.4.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.4.1.2. Tags

- clusterview.open-cluster-management.io : View a list of managed clusters that your ID can access.

1.7.4.2. Paths

1.7.4.2.1. Get managed clusters

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

1.7.4.2.1.1. Description

View a list of the managed clusters that you can access.

1.7.4.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.4.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.4.2.1.4. Consumes

- **managedcluster/yaml**

1.7.4.2.1.5. Tags

- clusterview.open-cluster-management.io

1.7.4.2.2. List managed clusters

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

1.7.4.2.2.1. Description

View a list of the managed clusters that you can access.

1.7.4.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *optional*	Name of the user ID for which you want to list the managed clusters.	string

1.7.4.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.4.2.2.4. Consumes

- **managedcluster/yaml**

1.7.4.2.2.5. Tags

- clusterview.open-cluster-management.io

1.7.4.2.2.6. Example HTTP request

1.7.4.2.2.6.1. Request body

```
{
  "apiVersion" : "clusterview.open-cluster-management.io/v1alpha1",
  "kind" : "ClusterView",
```

```
  "metadata" : {
    "name" : "<user_ID>"
  },
  "spec": { },
  "status" : { }
}
```

1.7.4.2.3. Watch the managed cluster sets

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

1.7.4.2.3.1. Description

Watch the managed clusters that you can access.

1.7.4.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

1.7.4.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.4.2.4. List the managed cluster sets.

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

1.7.4.2.4.1. Description

List the managed clusters that you can access.

1.7.4.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

1.7.4.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.4.2.5. List the managed cluster sets.

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

1.7.4.2.5.1. Description

List the managed clusters that you can access.

1.7.4.2.5.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

1.7.4.2.5.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.4.2.6. Watch the managed cluster sets.

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

1.7.4.2.6.1. Description

Watch the managed clusters that you can access.

1.7.4.2.6.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**clusterview_name** *optional*	Name of the user ID that you want to watch.	string

1.7.4.2.6.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.5. Managed service account (Technology Preview)

1.7.5.1. Overview

This documentation is for the **ManagedServiceAccount** resource for the multicluster engine for Kubernetes operator. The **ManagedServiceAccount** resource has four possible requests: create, query, delete, and update.

1.7.5.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.5.1.2. Tags

- **managedserviceaccounts.multicluster.openshift.io`** : Create and manage **ManagedServiceAccounts**

1.7.5.2. Paths

1.7.5.2.1. Create a ManagedServiceAccount

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

1.7.5.2.1.1. Description

Create a **ManagedServiceAccount**.

1.7.5.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the ManagedServiceAccount to be created.	ManagedServiceAccount

1.7.5.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content

HTTP Code	Description	Schema
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.5.2.1.4. Consumes

- **managedserviceaccount/yaml**

1.7.5.2.1.5. Tags

- managedserviceaccount.multicluster.openshift.io

1.7.5.2.1.5.1. Request body


```
{
  "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": []
  }
}
```

1.7.5.2.2. Query a single ManagedServiceAccount

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

1.7.5.2.2.1. Description

Query a single **ManagedServiceAccount** for more details.

1.7.5.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**managedserviceaccount_name** *required*	Name of the **ManagedServiceAccount** that you want to query.	string

1.7.5.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.5.2.2.4. Tags

- cluster.open-cluster-management.io

1.7.5.2.3. Delete a **ManagedServiceAccount**

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

1.7.5.2.3.1. Description

Delete a single **ManagedServiceAccount**.

1.7.5.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**managedserviceaccount_name** *required*	Name of the **ManagedServiceAccount** that you want to delete.	string

1.7.5.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.5.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.5.3. Definitions

1.7.5.3.1. **ManagedServiceAccount**

Name	Description	Schema
apiVersion *required*	The versioned schema of the **ManagedServiceAccount**.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of the **ManagedServiceAccount**.	object
spec *required*	The specification of the **ManagedServiceAccount**.	

1.7.6. MultiClusterEngine API

1.7.6.1. Overview

This documentation is for the MultiClusterEngine resource for multicluster engine for Kubernetes. The **MultiClusterEngine** resource has four possible requests: create, query, delete, and update.

1.7.6.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.6.1.2. Tags

- multiclusterengines.multicluster.openshift.io : Create and manage MultiClusterEngines

1.7.6.2. Paths

1.7.6.2.1. Create a MultiClusterEngine

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

1.7.6.2.1.1. Description

Create a MultiClusterEngine.

1.7.6.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Body	**body** *required*	Parameters describing the MultiClusterEngine to be created.	MultiClusterEngine

1.7.6.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.6.2.1.4. Consumes

- **MultiClusterEngines/yaml**

1.7.6.2.1.5. Tags

- multiclusterengines.multicluster.openshift.io

1.7.6.2.1.5.1. Request body


```
{
  "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": []
}
}
```


1.7.6.2.2. Query all MultiClusterEngines

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

1.7.6.2.2.1. Description

Query your multicluster engine for more details.

1.7.6.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.6.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.6.2.2.4. Consumes

- **operator/yaml**

1.7.6.2.2.5. Tags

- multiclusterengines.multicluster.openshift.io

1.7.6.2.3. Delete a MultiClusterEngine operator

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

1.7.6.2.3.1. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**name** *required*	Name of the multiclusterengine that you want to delete.	string

1.7.6.2.3.2. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.6.2.3.3. Tags

- multiclusterengines.multicluster.openshift.io

1.7.6.3. Definitions

1.7.6.3.1. MultiClusterEngine

Name	Description	Schema
apiVersion *required*	The versioned schema of the MultiClusterEngines.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	Describes rules that define the resource.	object
spec *required*	MultiClusterEngineSpec defines the desired state of MultiClusterEngine.	See *List of specs*

1.7.6.3.2. List of specs

Name	Description	Schema
nodeSelector *optional*	Set the nodeselectors.	map[string]string
imagePullSecret *optional*	Override pull secret for accessing MultiClusterEngine operand and endpoint images.	string

Name	Description	Schema
tolerations *optional*	Tolerations causes all components to tolerate any taints.	[]corev1.Toleration
targetNamespace *optional*	Location where MCE resources will be placed.	string

1.7.7. Placements API (v1beta1)

1.7.7.1. Overview

This documentation is for the Placement resource for multicluster engine for Kubernetes. Placement resource has four possible requests: create, query, delete and update.

1.7.7.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.7.1.2. Tags

- cluster.open-cluster-management.io : Create and manage Placements

1.7.7.2. Paths

1.7.7.2.1. Query all Placements

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

1.7.7.2.1.1. Description

Query your Placements for more details.

1.7.7.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.7.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.7.2.1.4. Consumes

- **placement/yaml**

1.7.7.2.1.5. Tags

- cluster.open-cluster-management.io

1.7.7.2.2. Create a Placement

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

1.7.7.2.2.1. Description

Create a Placement.

1.7.7.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the placement to be created.	Placement

1.7.7.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content

HTTP Code	Description	Schema
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.7.2.2.4. Consumes

- **placement/yaml**

1.7.7.2.2.5. Tags

- cluster.open-cluster-management.io

1.7.7.2.2.6. Example HTTP request

1.7.7.2.2.6.1. Request body

```
{
  "apiVersion" : "cluster.open-cluster-management.io/v1beta1",
  "kind" : "Placement",
  "metadata" : {
    "name" : "placement1",
    "namespace": "ns1"
  },
  "spec": {
    "predicates": [
      {
        "requiredClusterSelector": {
          "labelSelector": {
            "matchLabels": {
              "vendor": "OpenShift"
            }
          }
        }
      }
    ]
  },
  "status" : { }
}
```

1.7.7.2.3. Query a single Placement

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

1.7.7.2.3.1. Description

Query a single Placement for more details.

1.7.7.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placement_name** *required*	Name of the Placement that you want to query.	string

1.7.7.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.7.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.7.2.4. Delete a Placement

```
DELETE /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placements/{placement_name}
```

1.7.7.2.4.1. Description

Delete a single Placement.

1.7.7.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

Type	Name	Description	Schema
Path	**placement_name** *required*	Name of the Placement that you want to delete.	string

1.7.7.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.7.2.4.4. Tags

- cluster.open-cluster-management.io

1.7.7.3. Definitions

1.7.7.3.1. Placement

Name	Description	Schema
apiVersion *required*	The versioned schema of the Placement.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of the Placement.	object
spec *required*	The specification of the Placement.	spec

spec

Name	Description	Schema
ClusterSets *optional*	A subset of ManagedClusterSets from which the ManagedClusters are selected. If it is empty, ManagedClusters is selected from the ManagedClusterSets that are bound to the Placement namespace. Otherwise, ManagedClusters are selected from the intersection of this subset and the ManagedClusterSets are bound to the placement namespace.	string array
numberOfClusters *optional*	The desired number of ManagedClusters to be selected.	integer (int32)
predicates *optional*	A subset of cluster predicates to select ManagedClusters. The conditional logic is *OR*.	clusterPredicate array

clusterPredicate

Name	Description	Schema
requiredClusterSelector *optional*	A cluster selector to select ManagedClusters with a label and cluster claim.	clusterSelector

clusterSelector

Name	Description	Schema
labelSelector *optional*	A selector of ManagedClusters by label.	object
claimSelector *optional*	A selector of ManagedClusters by claim.	clusterClaimSelector

clusterClaimSelector

Name	Description	Schema
matchExpressions *optional*	A subset of the cluster claim selector requirements. The conditional logic is *AND*.	< object > array

1.7.8. PlacementDecisions API (v1beta1)

1.7.8.1. Overview

This documentation is for the PlacementDecision resource for multicluster engine for Kubernetes. PlacementDecision resource has four possible requests: create, query, delete and update.

1.7.8.1.1. URI scheme

BasePath : /kubernetes/apis
Schemes : HTTPS

1.7.8.1.2. Tags

- cluster.open-cluster-management.io : Create and manage PlacementDecisions.

1.7.8.2. Paths

1.7.8.2.1. Query all PlacementDecisions

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

1.7.8.2.1.1. Description

Query your PlacementDecisions for more details.

1.7.8.2.1.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string

1.7.8.2.1.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.8.2.1.4. Consumes

- **placementdecision/yaml**

1.7.8.2.1.5. Tags

- cluster.open-cluster-management.io

1.7.8.2.2. Create a PlacementDecision

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

1.7.8.2.2.1. Description

Create a PlacementDecision.

1.7.8.2.2.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Body	**body** *required*	Parameters describing the PlacementDecision to be created.	PlacementDecision

1.7.8.2.2.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.8.2.2.4. Consumes

- **placementdecision/yaml**

1.7.8.2.2.5. Tags

- cluster.open-cluster-management.io

1.7.8.2.2.6. Example HTTP request

1.7.8.2.2.6.1. Request 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" : { }
}
```

1.7.8.2.3. Query a single PlacementDecision

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

1.7.8.2.3.1. Description

Query a single PlacementDecision for more details.

1.7.8.2.3.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placementdecision_name** *required*	Name of the PlacementDecision that you want to query.	string

1.7.8.2.3.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content

HTTP Code	Description	Schema
503	Service unavailable	No Content

1.7.8.2.3.4. Tags

- cluster.open-cluster-management.io

1.7.8.2.4. Delete a PlacementDecision

```
DELETE /cluster.open-cluster-management.io/v1beta1/namespaces/{namespace}/placementdecisions/{placementdecision_name}
```

1.7.8.2.4.1. Description

Delete a single PlacementDecision.

1.7.8.2.4.2. Parameters

Type	Name	Description	Schema
Header	**COOKIE** *required*	Authorization: Bearer {ACCESS_TOKEN} ; ACCESS_TOKEN is the user access token.	string
Path	**placementdecision_name** *required*	Name of the PlacementDecision that you want to delete.	string

1.7.8.2.4.3. Responses

HTTP Code	Description	Schema
200	Success	No Content
403	Access forbidden	No Content
404	Resource not found	No Content
500	Internal service error	No Content
503	Service unavailable	No Content

1.7.8.2.4.4. Tags

- cluster.open-cluster-management.io

1.7.8.3. Definitions

1.7.8.3.1. PlacementDecision

Name	Description	Schema
apiVersion *required*	The versioned schema of PlacementDecision.	string
kind *required*	String value that represents the REST resource.	string
metadata *required*	The meta data of PlacementDecision.	object

1.8. TROUBLESHOOTING

Before using the Troubleshooting guide, you can run the **oc adm must-gather** command to gather details, logs, and take steps in debugging issues. For more details, see Running the must-gather command to troubleshoot.

Additionally, check your role-based access. See Role-based access control for details.

1.8.1. Documented troubleshooting

View the list of troubleshooting topics for the multicluster engine for Kubernetes operator:

Installation

To view the main documentation for the installing tasks, see Install.

- Troubleshooting installation status stuck in installing or pending
- Troubleshooting reinstallation failure

Cluster management

To view the main documentation about managing your clusters, see The multicluster engine for Kubernetes operator cluster lifecycle overview.

- Troubleshooting an offline cluster
- Troubleshooting a managed cluster import failure
- Reimporting cluster fails with unknown authority error
- Troubleshooting cluster with pending import status
- Troubleshooting imported clusters offline after certificate change
- Troubleshooting cluster status changing from offline to available
- Troubleshooting cluster creation on VMware vSphere

- Troubleshooting cluster in console with pending or failed status
- Troubleshooting OpenShift Container Platform version 3.11 cluster import failure
- Troubleshooting Klusterlet with degraded conditions
- Namespace remains after deleting a cluster
- Auto-import-secret-exists error when importing a cluster

1.8.2. Running the must-gather command to troubleshoot

To get started with troubleshooting, learn about the troubleshooting scenarios for users to run the **must-gather** command to debug the issues, then see the procedures to start using the command.

Required access: Cluster administrator

1.8.2.1. Must-gather scenarios

- **Scenario one:** Use the *Documented troubleshooting* section to see if a solution to your problem is documented. The guide is organized by the major functions of the product.
  With this scenario, you check the guide to see if your solution is in the documentation.
- **Scenario two:** If your problem is not documented with steps to resolve, run the **must-gather** command and use the output to debug the issue.
- **Scenario three:** If you cannot debug the issue using your output from the **must-gather** command, then share your output with Red Hat Support.

1.8.2.2. Must-gather procedure

See the following procedure to start using the **must-gather** command:

1. Learn about the **must-gather** command and install the prerequisites that you need at Gathering data about your cluster in the RedHat OpenShift Container Platform documentation.
2. Log in to your cluster. For the usual use-case, you should run the **must-gather** while you are logged into your *engine* cluster.
   Note: If you want to check your managed clusters, find the **gather-managed.log** file that is located in the the **cluster-scoped-resources** directory:

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

   Check for managed clusters that are not set **True** for the JOINED and AVAILABLE column. You can run the **must-gather** command on those clusters that are not connected with **True** status.
3. Add the multicluster engine for Kubernetes image that is used for gathering data and the directory. Run the following command, where you insert the image and the directory for the output:

   ```
   oc adm must-gather --image=registry.redhat.io/multicluster-engine/must-gather-rhel8:v2.1.0 --dest-dir=<directory>
   ```
4. Go to your specified directory to see your output, which is organized in the following levels:

- Two peer levels: **cluster-scoped-resources** and **namespace** resources.
- Sub-level for each: API group for the custom resource definitions for both cluster-scope and namespace-scoped resources.
- Next level for each: YAML file sorted by **kind**.

1.8.2.3. Must-gather in a disconnected environment

Complete the following steps to run the **must-gather** command in a disconnected environment:

1. In a disconnected environment, mirror the Red Hat operator catalog images into their mirror registry. For more information, see Install on disconnected networks.
2. Run the following command to extract logs, which reference the image from their mirror registry:

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

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

You can open a bug for the product team here.

- Running the must-gather command to troubleshoot

1.8.3. Troubleshooting installation status stuck in installing or pending

When installing the multicluster engine for Kubernetes operator, the **MultiClusterEngine** remains in **Installing** phase, or multiple pods maintain a **Pending** status.

1.8.3.1. Symptom: Stuck in Pending status

More than ten minutes passed since you installed **MultiClusterEngine** and one or more components from the **status.components** field of the **MultiClusterEngine** resource report **ProgressDeadlineExceeded**. Resource constraints on the cluster might be the issue.

Check the pods in the namespace where **MultiClusterEngine** was installed. You might see **Pending** with a status similar to the following:

```
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.'
```

In this case, the worker nodes resources are not sufficient in the cluster to run the product.

1.8.3.2. Resolving the problem: Adjust worker node sizing

If you have this problem, then your cluster needs to be updated with either larger or more worker nodes. See Sizing your cluster for guidelines on sizing your cluster.

1.8.4. Troubleshooting reinstallation failure

When reinstalling multicluster engine for Kubernetes operator, the pods do not start.

1.8.4.1. Symptom: Reinstallation failure

If your pods do not start after you install the multicluster engine for Kubernetes operator, it is often because items from a previous installation of multicluster engine for Kubernetes operator were not removed correctly when it was uninstalled.

In this case, the pods do not start after completing the installation process.

1.8.4.2. Resolving the problem: Reinstallation failure

If you have this problem, complete the following steps:

1. Run the uninstallation process to remove the current components by following the steps in Uninstalling.
2. Install the Helm CLI binary version 3.2.0, or later, by following the instructions at Installing Helm.
3. Ensure that your Red Hat OpenShift Container Platform CLI is configured to run **oc** commands. See Getting started with the OpenShift CLI in the OpenShift Container Platform documentation for more information about how to configure the **oc** commands.
4. Copy the following script into a file:

   ```
   #!/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
   ```

   Replace **<namespace>** in the script with the name of the namespace where multicluster engine for Kubernetes operator was installed. Ensure that you specify the correct namespace, as the namespace is cleaned out and deleted.
5. Run the script to remove the artifacts from the previous installation.
6. Run the installation. See Installing while connected online .

1.8.5. Troubleshooting an offline cluster

There are a few common causes for a cluster showing an offline status.

1.8.5.1. Symptom: Cluster status is offline

After you complete the procedure for creating a cluster, you cannot access it from the Red Hat Advanced Cluster Management console, and it shows a status of **offline**.

1.8.5.2. Resolving the problem: Cluster status is offline

1. Determine if the managed cluster is available. You can check this in the *Clusters* area of the Red Hat Advanced Cluster Management console.
   If it is not available, try restarting the managed cluster.

2. If the managed cluster status is still offline, complete the following steps:

   a. Run the **oc get managedcluster <cluster_name> -o yaml** command on the hub cluster. Replace **<cluster_name>** with the name of your cluster.

   b. Find the **status.conditions** section.

   c. Check the messages for **type: ManagedClusterConditionAvailable** and resolve any problems.

1.8.6. Troubleshooting a managed cluster import failure

If your cluster import fails, there are a few steps that you can take to determine why the cluster import failed.

1.8.6.1. Symptom: Imported cluster not available

After you complete the procedure for importing a cluster, you cannot access it from the console.

1.8.6.2. Resolving the problem: Imported cluster not available

There can be a few reasons why an imported cluster is not available after an attempt to import it. If the cluster import fails, complete the following steps, until you find the reason for the failed import:

1. On the hub cluster, run the following command to ensure that the import controller is running.

   ```
   kubectl -n multicluster-engine get pods -l app=managedcluster-import-controller-v2
   ```

   You should see two pods that are running. If either of the pods is not running, run the following command to view the log to determine the reason:

   ```
   kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1
   ```

2. On the hub cluster, run the following command to determine if the managed cluster import secret was generated successfully by the import controller:

   ```
   kubectl -n <managed_cluster_name> get secrets <managed_cluster_name>-import
   ```

   If the import secret does not exist, run the following command to view the log entries for the import controller and determine why it was not created:

   ```
   kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 --tail=-1 | grep importconfig-controller
   ```

3. On the hub cluster, if your managed cluster is **local-cluster**, provisioned by Hive, or has an auto-import secret, run the following command to check the import status of the managed cluster.

   ```
   kubectl get managedcluster <managed_cluster_name> -o=jsonpath='{range .status.conditions[*]}{.type}{"\t"}{.status}{"\t"}{.message}{"\n"}{end}' | grep ManagedClusterImportSucceeded
   ```

-

If the condition **ManagedClusterImportSucceeded** is not **true**, the result of the command indicates the reason for the failure.

4. Check the Klusterlet status of the managed cluster for a degraded condition. See Troubleshooting Klusterlet with degraded conditions to find the reason that the Klusterlet is degraded.

1.8.7. Reimporting cluster fails with unknown authority error

If you experience a problem when reimporting a managed cluster to your multicluster engine for Kubernetes operator hub cluster, follow the procedure to troubleshoot the problem.

1.8.7.1. Symptom: Reimporting cluster fails with unknown authority error

After you provision an OpenShift Container Platform cluster with multicluster engine for Kubernetes operator, reimporting the cluster might fail with a **x509: certificate signed by unknown authority** error when you change or add API server certificates to your OpenShift Container Platform cluster.

1.8.7.2. Identifying the problem: Reimporting cluster fails with unknown authority error

After failing to reimport your managed cluster, run the following command to get the import controller log on your multicluster engine for Kubernetes operator hub cluster:

```
kubectl -n multicluster-engine logs -l app=managedcluster-import-controller-v2 -f
```

If the following error log appears, your managed cluster API server certificates might have changed:

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"}

To determine if your managed cluster API server certificates have changed, complete the following steps:

1. Run the following command to specify your managed cluster name by replacing **your-managed-cluster-name** with the name of your managed cluster:

```
cluster_name=<your-managed-cluster-name>
```

2. Get your managed cluster **kubeconfig** secret name by running the following command:

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

3. Export **kubeconfig** to a new file by running the following commands:

```
oc -n ${cluster_name} get secret ${kubeconfig_secret_name} -ojsonpath={.data.kubeconfig} | base64 -d > kubeconfig.old
```

```
export KUBECONFIG=kubeconfig.old
```

4. Get the namespace from your managed cluster with **kubeconfig** by running the following command:

```
oc get ns
```

If you receive an error that resembles the following message, your cluster API server ceritificates have been changed and your **kubeconfig** file is invalid.

Unable to connect to the server: x509: certificate signed by unknown authority

1.8.7.3. Resolving the problem: Reimporting cluster fails with unknown authority error

The managed cluster administrator must create a new valid **kubeconfig** file for your managed cluster.

After creating a new **kubeconfig**, complete the following steps to update the new **kubeconfig** for your managed cluster:

1. Run the following commands to set your **kubeconfig** file path and cluster name. Replace **<path_to_kubeconfig>** with the path to your new **kubeconfig** file. Replace **<managed_cluster_name>** with the name of your managed cluster:

```
cluster_name=<managed_cluster_name>
kubeconfig_file=<path_to_kubeconfig>
```

2. Run the following command to encode your new **kubeconfig**:

```
kubeconfig=$(cat ${kubeconfig_file} | base64 -w0)
```

Note: On macOS, run the following command instead:

```
kubeconfig=$(cat ${kubeconfig_file} | base64)
```

3. Run the following command to define the **kubeconfig** json patch:

```
kubeconfig_patch="[\{\"op\":\"replace\", \"path\":\"/data/kubeconfig\",
\"value\":\"${kubeconfig}\"}, \{\"op\":\"replace\", \"path\":\"/data/raw-kubeconfig\",
\"value\":\"${kubeconfig}\"}]"
```

4. Retrieve your administrator **kubeconfig** secret name from your managed cluster by running the following command:

```
kubeconfig_secret_name=$(oc -n ${cluster_name} get clusterdeployments ${cluster_name} -
ojsonpath='{.spec.clusterMetadata.adminKubeconfigSecretRef.name}')
```

5. Patch your administrator **kubeconfig** secret with your new **kubeconfig** by running the following command:

```
oc -n ${cluster_name} patch secrets ${kubeconfig_secret_name} --type='json' -
p="${kubeconfig_patch}"
```

1.8.8. Troubleshooting cluster with pending import status

If you receive *Pending import* continually on the console of your cluster, follow the procedure to troubleshoot the problem.

1.8.8.1. Symptom: Cluster with pending import status

After importing a cluster by using the Red Hat Advanced Cluster Management console, the cluster appears in the console with a status of *Pending import*.

1.8.8.2. Identifying the problem: Cluster with pending import status

1. Run the following command on the managed cluster to view the Kubernetes pod names that are having the issue:

   ```
   kubectl get pod -n open-cluster-management-agent | grep klusterlet-registration-agent
   ```

2. Run the following command on the managed cluster to find the log entry for the error:

   ```
   kubectl logs <registration_agent_pod> -n open-cluster-management-agent
   ```

   Replace *registration_agent_pod* with the pod name that you identified in step 1.

3. Search the returned results for text that indicates there was a networking connectivity problem. Example includes: **no such host**.

1.8.8.3. Resolving the problem: Cluster with pending import status

1. Retrieve the port number that is having the problem by entering the following command on the hub cluster:

   ```
   oc get infrastructure cluster -o yaml | grep apiServerURL
   ```

2. Ensure that the hostname from the managed cluster can be resolved, and that outbound connectivity to the host and port is occurring.
   If the communication cannot be established by the managed cluster, the cluster import is not complete. The cluster status for the managed cluster is *Pending import*.

1.8.9. Troubleshooting imported clusters offline after certificate change

Installing a custom **apiserver** certificate is supported, but one or more clusters that were imported before you changed the certificate information can have an **offline** status.

1.8.9.1. Symptom: Clusters offline after certificate change

After you complete the procedure for updating a certificate secret, one or more of your clusters that were online are now displaying an **offline** status in the console.

1.8.9.2. Identifying the problem: Clusters offline after certificate change

After updating the information for a custom API server certificate, clusters that were imported and running before the new certificate are now in an **offline** state.

The errors that indicate that the certificate is the problem are found in the logs for the pods in the **open-cluster-management-agent** namespace of the offline managed cluster. The following examples are similar to the errors that are displayed in the logs:

Log of **work-agent**:

```
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
```

Log of **registration-agent**:

```
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
```

1.8.9.3. Resolving the problem: Clusters offline after certificate change

If your managed cluster is the **local-cluster** or your managed cluster was created by using multicluster engine for Kubernetes operator, you must wait 10 minutes or longer to reimport your managed cluster.

To reimport your managed cluster immediately, you can delete your managed cluster import secret on the hub cluster and reimport it by using {mce-short}. Run the following command:

```
oc delete secret -n <cluster_name> <cluster_name>-import
```

Replace **<cluster_name>** with the name of the managed cluster that you want to import.

If you want to reimport a managed cluster that was imported by using {mce-short}, complete the following steps import the managed cluster again:

1. On the hub cluster, recreate the managed cluster import secret by running the following command:

   ```
   oc delete secret -n <cluster_name> <cluster_name>-import
   ```

   Replace **<cluster_name>** with the name of the managed cluster that you want to import.

2. On the hub cluster, expose the managed cluster import secret to a YAML file by running the following command:

   ```
   oc get secret -n <cluster_name> <cluster_name>-import -ojsonpath='{.data.import\.yaml}' |
   base64 --decode  > import.yaml
   ```

   Replace **<cluster_name>** with the name of the managed cluster that you want to import.

3. On the managed cluster, apply the **import.yaml** fileby running the following command:

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

Note: The previous steps do not detach the managed cluster from the hub cluster. The steps update the required manifests with current settings on the managed cluster, including the new certificate information.

1.8.10. Troubleshooting cluster status changing from offline to available

The status of the managed cluster alternates between **offline** and **available** without any manual change to the environment or cluster.

1.8.10.1. Symptom: Cluster status changing from offline to available

When the network that connects the managed cluster to the hub cluster is unstable, the status of the managed cluster that is reported by the hub cluster cycles between **offline** and **available**.

1.8.10.2. Resolving the problem: Cluster status changing from offline to available

To attempt to resolve this issue, complete the following steps:

1. Edit your **ManagedCluster** specification on the hub cluster by entering the following command:

   ```
   oc edit managedcluster <cluster-name>
   ```

   Replace *cluster-name* with the name of your managed cluster.

2. Increase the value of **leaseDurationSeconds** in your **ManagedCluster** specification. The default value is 5 minutes, but that might not be enough time to maintain the connection with the network issues. Specify a greater amount of time for the lease. For example, you can raise the setting to 20 minutes.

1.8.11. Troubleshooting cluster creation on VMware vSphere

If you experience a problem when creating a Red Hat OpenShift Container Platform cluster on VMware vSphere, see the following troubleshooting information to see if one of them addresses your problem.

Note: Sometimes when the cluster creation process fails on VMware vSphere, the link is not enabled for you to view the logs. If this happens, you can identify the problem by viewing the log of the **hive-controllers** pod. The **hive-controllers** log is in the **hive** namespace.

1.8.11.1. Managed cluster creation fails with certificate IP SAN error

1.8.11.1.1. Symptom: Managed cluster creation fails with certificate IP SAN error

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails with an error message that indicates a certificate IP SAN error.

1.8.11.1.2. Identifying the problem: Managed cluster creation fails with certificate IP SAN error

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
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
```

1.8.11.1.3. Resolving the problem: Managed cluster creation fails with certificate IP SAN error

Use the VMware vCenter server fully-qualified host name instead of the IP address in the credential. You can also update the VMware vCenter CA certificate to contain the IP SAN.

1.8.11.2. Managed cluster creation fails with unknown certificate authority

1.8.11.2.1. Symptom: Managed cluster creation fails with unknown certificate authority

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because the certificate is signed by an unknown authority.

1.8.11.2.2. Identifying the problem: Managed cluster creation fails with unknown certificate authority

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
Error: error setting up new vSphere SOAP client: Post https://vspherehost.com/sdk: x509: certificate signed by unknown authority"
```

1.8.11.2.3. Resolving the problem: Managed cluster creation fails with unknown certificate authority

Ensure you entered the correct certificate from the certificate authority when creating the credential.

1.8.11.3. Managed cluster creation fails with expired certificate

1.8.11.3.1. Symptom: Managed cluster creation fails with expired certificate

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because the certificate is expired or is not yet valid.

1.8.11.3.2. Identifying the problem: Managed cluster creation fails with expired certificate

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
x509: certificate has expired or is not yet valid
```

1.8.11.3.3. Resolving the problem: Managed cluster creation fails with expired certificate

Ensure that the time on your ESXi hosts is synchronized.

1.8.11.4. Managed cluster creation fails with insufficient privilege for tagging

1.8.11.4.1. Symptom: Managed cluster creation fails with insufficient privilege for tagging

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is insufficient privilege to use tagging.

1.8.11.4.2. Identifying the problem: Managed cluster creation fails with insufficient privilege for tagging

The deployment of the managed cluster fails and returns the following errors in the deployment log:

```
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\"
{"
```

1.8.11.4.3. Resolving the problem: Managed cluster creation fails with insufficient privilege for tagging

Ensure that your VMware vCenter required account privileges are correct. See Image registry removed during information for more information.

1.8.11.5. Managed cluster creation fails with invalid dnsVIP

1.8.11.5.1. Symptom: Managed cluster creation fails with invalid dnsVIP

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an invalid dnsVIP.

1.8.11.5.2. Identifying the problem: Managed cluster creation fails with invalid dnsVIP

If you see the following message when trying to deploy a new managed cluster with VMware vSphere, it is because you have an older OpenShift Container Platform release image that does not support VMware Installer Provisioned Infrastructure (IPI):

```
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
```

1.8.11.5.3. Resolving the problem: Managed cluster creation fails with invalid dnsVIP

Select a release image from a later version of OpenShift Container Platform that supports VMware Installer Provisioned Infrastructure.

1.8.11.6. Managed cluster creation fails with incorrect network type

1.8.11.6.1. Symptom: Managed cluster creation fails with incorrect network type

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an incorrect network type specified.

1.8.11.6.2. Identifying the problem: Managed cluster creation fails with incorrect network type

If you see the following message when trying to deploy a new managed cluster with VMware vSphere, it is because you have an older OpenShift Container Platform image that does not support VMware Installer Provisioned Infrastructure (IPI):

```
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"
```

1.8.11.6.3. Resolving the problem: Managed cluster creation fails with incorrect network type

Select a valid VMware vSphere network type for the specified VMware cluster.

1.8.11.7. Managed cluster creation fails with an error processing disk changes

1.8.11.7.1. Symptom: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

After creating a new Red Hat OpenShift Container Platform cluster on VMware vSphere, the cluster fails because there is an error when processing disk changes.

1.8.11.7.2. Identifying the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

A message similar to the following is displayed in the logs:

```
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)
```

1.8.11.7.3. Resolving the problem: Adding the VMware vSphere managed cluster fails due to an error processing disk changes

Use the VMware vSphere client to give the user **All privileges** for *Profile-driven Storage Privileges*.

1.8.12. Troubleshooting cluster in console with pending or failed status

If you observe *Pending* status or *Failed* status in the console for a cluster you created, follow the procedure to troubleshoot the problem.

1.8.12.1. Symptom: Cluster in console with pending or failed status

After creating a new cluster by using the console, the cluster does not progress beyond the status of *Pending* or displays *Failed* status.

1.8.12.2. Identifying the problem: Cluster in console with pending or failed status

If the cluster displays *Failed* status, navigate to the details page for the cluster and follow the link to the logs provided. If no logs are found or the cluster displays *Pending* status, continue with the following procedure to check for logs:

- Procedure 1

  1. Run the following command on the hub cluster to view the names of the Kubernetes pods that were created in the namespace for the new cluster:

     ```
     oc get pod -n <new_cluster_name>
     ```

     Replace ***new_cluster_name*** with the name of the cluster that you created.

  2. If no pod that contains the string **provision** in the name is listed, continue with Procedure 2. If there is a pod with **provision** in the title, run the following command on the hub cluster to view the logs of that pod:

     ```
     oc logs <new_cluster_name_provision_pod_name> -n <new_cluster_name> -c hive
     ```

     Replace **new_cluster_name_provision_pod_name** with the name of the cluster that you created, followed by the pod name that contains **provision**.

  3. Search for errors in the logs that might explain the cause of the problem.

- Procedure 2
  If there is not a pod with **provision** in its name, the problem occurred earlier in the process. Complete the following procedure to view the logs:

  1. Run the following command on the hub cluster:

     ```
     oc describe clusterdeployments -n <new_cluster_name>
     ```

     Replace **new_cluster_name** with the name of the cluster that you created. For more information about cluster installation logs, see Gathering installation logs in the Red Hat OpenShift documentation.

  2. See if there is additional information about the problem in the *Status.Conditions.Message* and *Status.Conditions.Reason* entries of the resource.

1.8.12.3. Resolving the problem: Cluster in console with pending or failed status

After you identify the errors in the logs, determine how to resolve the errors before you destroy the cluster and create it again.

The following example provides a possible log error of selecting an unsupported zone, and the actions that are required to resolve it:

```
No subnets provided for zones
```

When you created your cluster, you selected one or more zones within a region that are not supported. Complete one of the following actions when you recreate your cluster to resolve the issue:

- Select a different zone within the region.
- Omit the zone that does not provide the support, if you have other zones listed.
- Select a different region for your cluster.

After determining the issues from the log, destroy the cluster and recreate it.

See Creating a cluster for more information about creating a cluster.

1.8.13. Troubleshooting OpenShift Container Platform version 3.11 cluster import failure

1.8.13.1. Symptom: OpenShift Container Platform version 3.11 cluster import failure

After you attempt to import a Red Hat OpenShift Container Platform version 3.11 cluster, the import fails with a log message that resembles the following content:

```
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|...
```

1.8.13.2. Identifying the problem: OpenShift Container Platform version 3.11 cluster import failure

This often occurs because the installed version of the **kubectl** command-line tool is 1.11, or earlier. Run the following command to see which version of the **kubectl** command-line tool you are running:

```
kubectl version
```

If the returned data lists version 1.11, or earlier, complete one of the fixes in *Resolving the problem: OpenShift Container Platform version 3.11 cluster import failure*.

1.8.13.3. Resolving the problem: OpenShift Container Platform version 3.11 cluster import failure

You can resolve this issue by completing one of the following procedures:

- Install the latest version of the **kubectl** command-line tool.
  1. Download the latest version of the **kubectl** tool from: Install and Set Up kubectl in the Kubernetes documentation.
  2. Import the cluster again after upgrading your **kubectl** tool.
- Run a file that contains the import command.
  1. Start the procedure in Importing a managed cluster with the CLI.
  2. When you create the command to import your cluster, copy that command into a YAML file named **import.yaml**.
  3. Run the following command to import the cluster again from the file:

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

1.8.14. Troubleshooting Klusterlet with degraded conditions

The Klusterlet degraded conditions can help to diagnose the status of Klusterlet agents on managed cluster. If a Klusterlet is in the degraded condition, the Klusterlet agents on managed cluster might have errors that need to be troubleshooted. See the following information for Klusterlet degraded conditions that are set to **True**.

1.8.14.1. Symptom: Klusterlet is in the degraded condition

After deploying a Klusterlet on managed cluster, the **KlusterletRegistrationDegraded** or **KlusterletWorkDegraded** condition displays a status of *True*.

1.8.14.2. Identifying the problem: Klusterlet is in the degraded condition

1. Run the following command on the managed cluster to view the Klusterlet status:

   ```
   kubectl get klusterlets klusterlet -oyaml
   ```

2. Check **KlusterletRegistrationDegraded** or **KlusterletWorkDegraded** to see if the condition is set to **True**. Proceed to *Resolving the problem* for any degraded conditions that are listed.

1.8.14.3. Resolving the problem: Klusterlet is in the degraded condition

See the following list of degraded statuses and how you can attempt to resolve those issues:

- If the **KlusterletRegistrationDegraded** condition with a status of *True* and the condition reason is: *BootStrapSecretMissing*, you need create a bootstrap secret on **open-cluster-management-agent** namespace.

- If the **KlusterletRegistrationDegraded** condition displays *True* and the condition reason is a *BootstrapSecretError*, or *BootstrapSecretUnauthorized*, then the current bootstrap secret is invalid. Delete the current bootstrap secret and recreate a valid bootstrap secret on **open-cluster-management-agent** namespace.

- If the **KlusterletRegistrationDegraded** and **KlusterletWorkDegraded** displays *True* and the condition reason is *HubKubeConfigSecretMissing*, delete the Klusterlet and recreate it.

- If the **KlusterletRegistrationDegraded** and **KlusterletWorkDegraded** displays *True* and the condition reason is: *ClusterNameMissing*, *KubeConfigMissing*, *HubConfigSecretError*, or *HubConfigSecretUnauthorized*, delete the hub cluster kubeconfig secret from **open-cluster-management-agent** namespace. The registration agent will bootstrap again to get a new hub cluster kubeconfig secret.

- If the **KlusterletRegistrationDegraded** displays *True* and the condition reason is *GetRegistrationDeploymentFailed* or *UnavailableRegistrationPod*, you can check the condition message to get the problem details and attempt to resolve.

- If the **KlusterletWorkDegraded** displays *True* and the condition reason is *GetWorkDeploymentFailed* or *UnavailableWorkPod*, you can check the condition message to get the problem details and attempt to resolve.

1.8.15. Namespace remains after deleting a cluster

When you remove a managed cluster, the namespace is normally removed as part of the cluster removal process. In rare cases, the namespace remains with some artifacts in it. In that case, you must manually remove the namespace.

1.8.15.1. Symptom: Namespace remains after deleting a cluster

After removing a managed cluster, the namespace is not removed.

1.8.15.2. Resolving the problem: Namespace remains after deleting a cluster

Complete the following steps to remove the namespace manually:

1. Run the following command to produce a list of the resources that remain in the <cluster_name> namespace:

   ```
   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>
   ```

   Replace **cluster_name** with the name of the namespace for the cluster that you attempted to remove.

2. Delete each identified resource on the list that does not have a status of **Delete** by entering the following command to edit the list:

   ```
   oc edit <resource_kind> <resource_name> -n <namespace>
   ```

Replace **resource_kind** with the kind of the resource. Replace **resource_name** with the name of the resource. Replace **namespace** with the name of the namespace of the resource.

3. Locate the **finalizer** attribute in the in the metadata.
4. Delete the non-Kubernetes finalizers by using the vi editor **dd** command.
5. Save the list and exit the **vi** editor by entering the **:wq** command.
6. Delete the namespace by entering the following command:

```
oc delete ns <cluster-name>
```

Replace **cluster-name** with the name of the namespace that you are trying to delete.

1.8.16. Auto-import-secret-exists error when importing a cluster

Your cluster import fails with an error message that reads: auto import secret exists.

1.8.16.1. Symptom: Auto import secret exists error when importing a cluster

When importing a hive cluster for management, an **auto-import-secret already exists** error is displayed.

1.8.16.2. Resolving the problem: Auto-import-secret-exists error when importing a cluster

This problem occurs when you attempt to import a cluster that was previously managed. When this happens, the secrets conflict when you try to reimport the cluster.

To work around this problem, complete the following steps:

1. To manually delete the existing **auto-import-secret**, run the following command on the hub cluster:

```
oc delete secret auto-import-secret -n <cluster-namespace>
```

Replace **cluster-namespace** with the namespace of your cluster.

2. Import your cluster again using the procedure in Importing a target managed cluster to a hub cluster.

Your cluster is imported.