Red Hat Advanced Cluster Management for Kubernetes 2.9

Multicluster global hub

Multicluster global hub

Last Updated: 2024-04-15

Red Hat Advanced Cluster Management for Kubernetes 2.9 Multicluster global hub

Multicluster global hub

Legal Notice

Abstract

Read more to learn about the multicluster global hub, which enables you to manage multiple hub clusters with a single hub.

Table of Contents

CHAPTER 1. MULTICLUSTER GLOBAL HUB

The multicluster global hub is a set of components that enable you to import one or more hub clusters and manage them from a single hub cluster.

After importing the hub clusters as managed hub clusters, you can use multicluster global hub to complete the following tasks across all of the managed hub clusters:

- Report the policy compliance status and trend
- Inventory all managed hubs and managed clusters on the overview page
- Detect and alert in cases of irregular policy behavior

The multicluster global hub is useful when a single hub cluster cannot manage the large number of clusters in a high-scale environment. When this happens, you divide the clusters into smaller groups of clusters and configure a hub cluster for each group.

It is often inconvenient to view the data on multiple hub clusters for the managed clusters that are managed by that hub cluster. The multicluster global hub provides an easier way to view information from multiple hubs by designating multiple hub clusters as managed hub clusters. The multicluster global hub cluster manages the other hub clusters and gathers summarized information from the managed hub clusters.

- multicluster global hub architecture
- Global Hub requirements
- Installing Multicluster Global Hub in a connected environment
- Installing Multicluster Global Hub in a disconnected environment
- Integrating existing components
- Importing a managed hub cluster in the default mode
- Accessing the Grafana data
- Grafana alerts (Technology Preview)
- Configuring the cron jobs
- Running the summarization process manually

1.1. MULTICLUSTER GLOBAL HUB ARCHITECTURE

The multicluster global hub consists of the following components that are used to access and manage your hub clusters:

- A server component called the *global hub cluster* where the management tools and the console run
- A client component that is installed on Red Hat Advanced Cluster Management, named the *managed hub*, which can be managed by the global hub cluster. The managed hub also manages other clusters. You do not have to use a dedicated cluster for your multicluster global hub cluster.

Learn more about the architecture in the following sections:

See the following high-level multicluster terms and components:

- multicluster global hub operator
- multicluster global hub manager
- multicluster global hub agent
- multicluster global hub visualizations

1.1.1. The multicluster global hub operator

The multicluster global hub operator contains the components of multicluster global hub. The operator deploys all of the required components for global multicluster management. The components include **multicluster-global-hub-manager**, **multicluster-global-hub-grafana**, and provided versions of **Kafka** and **PostgreSQL** in the multicluster global hub cluster and **multicluster-global-hub-agent** in the managed hub clusters.

The operator also leverages the **manifestwork** custom resource to deploy the Red Hat Advanced Cluster Management for Kubernetes operator on the managed cluster. After the Red Hat Advanced Cluster Management operator is deployed on the managed cluster, the managed cluster becomes a standard Red Hat Advanced Cluster Management cluster. This hub cluster is now a managed hub cluster.

1.1.2. The multicluster global hub manager

The multicluster global hub manager is used to persist the data into the **postgreSQL** database. The data is from Kafka transport. The manager also posts the data to the Kafka transport, so it can be synchronized with the data on the managed hub clusters.

1.1.3. The multicluster global hub agent

The multicluster global hub agent runs on the managed hub clusters. It synchronizes the data between the multicluster global hub cluster and the managed hub clusters. For example, the agent synchronizes the information of the managed clusters from the managed hub clusters to the multicluster global hub cluster and synchronizes the policy or application from the multicluster global hub cluster to the managed hub clusters.

1.1.4. The multicluster global hub visualizations

Grafana runs on the multicluster global hub cluster as the main service for multicluster global hub visualizations. The PostgreSQL data collected by the Global Hub Manager is its default DataSource. By exposing the service using the route called **multicluster-global-hub-grafana**, you can access the multicluster global hub Grafana dashboards by accessing the console.

1.2. GLOBAL HUB REQUIREMENTS

Learn about what is required for installation and networking, as well as supported components and environments.

- General requirements
- Networking requirements

- Supported components

1.2.1. General requirements

To install Global hub, you need the following requirements:

Required access: Cluster administrator

OpenShift Container Platform Dedicated environment required access: 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.

- Red Hat Advanced Cluster Management for Kubernetes must be installed and configured. Learn more details about Red Hat Advanced Cluster Management .

1.2.2. Networking requirements

See the following networking requirements:

- The managed hub is also a managed cluster of multicluster global hub in Red Hat Advanced Cluster Management. The network configuration in Red Hat Advanced Cluster Management is necessary. See Networking for Red Hat Advanced Cluster Management networking details.
- The following table lists the Global hub network information:

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address	
Inbound from browser of the user	HTTPS	User need to access the Grafana dashboard	443	Browser of the user	IP address of Grafana route	
Outbound to Kafka Cluster	HTTPS	Global hub manager need to get data from Kafka cluster	443	multicluster-global-hub-manager-xxx pod	Kafka route host	
Outbound to PostgreSQL database	HTTPS	Global hub manager need to persist data to PostgreSQL database	443	multicluster-global-hub-manager-xxx pod	IP address of the PostgreSQL database	

- The following table lists the Managed hub network information:

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address	
Outbound to Kafka Cluster	HTTPS	Global hub agent need to sync cluster info and policy info to Kafka cluster	443	multicluster-global-hub-agent pod	Kafka route host	

- You can see guidelines for sizing at Sizing your Red Hat Advanced Cluster Management cluster in the product documentation.
- **Optional:** For middleware, multicluster global hub has built-in Kafka, PostgreSQL, and Grafana, but you can use your own configured Kafka, PostgreSQL, and Grafana. See Integrating existing components for more details.

1.2.3. Supported components

Learn about supported platforms and components.

- Because they share an integrated console, the multicluster global hub console supports the same browsers as the OpenShift Container Platform. See Accessing the web console in the Red Hat OpenShift Container Platform documentation for information about supported browsers and versions.
- The platforms available for the supported multicluster global hub cluster are shown in the following table:

Platform	Supported for global hub cluster	Supported for managed hub cluster
Red Hat Advanced Cluster Management 2.9, and later 2.9.x releases	Yes	Yes
Red Hat Advanced Cluster Management 2.8.3, and later 2.8.x releases	Yes	Yes
Red Hat Advanced Cluster Management 2.7.9, and later 2.7.x releases	Yes	Yes
Red Hat Advanced Cluster Management on Arm	No	Yes

Platform	Supported for global hub cluster	Supported for managed hub cluster
Red Hat Advanced Cluster Management on IBM Z	No	Yes
Red Hat Advanced Cluster Management on IBM Power Systems	No	Yes

- The multicluster global hub supports the following middleware:
  - Kafka 3.3 and later 3.3.x releases.
  - PostgreSQL version 13 and later 13.x releases.

1.2.4. Additional resources

- Installing Multicluster Global Hub in a connected environment
- Installing Multicluster Global Hub in a disconnected environment

1.3. INSTALLING MULTICLUSTER GLOBAL HUB IN A CONNECTED ENVIRONMENT

The multicluster global hub is installed through Operator Lifecycle Manager, which manages the installation, upgrade, and removal of the components that comprise the operator.

Required access: Cluster administrator

1.3.1. Prerequisites

- For the OpenShift Container Platform Dedicated environment, you must have **cluster-admin** permissions to access the environment. By default **dedicated-admin** role does not have the required permissions to create namespaces in the OpenShift Container Platform Dedicated environment.
- You must install and configure Red Hat Advanced Cluster Management for Kubernetes. For more details, see Installing and upgrading .
- You must configure the Red Hat Advanced Cluster Management network. The managed hub cluster is also a managed cluster of multicluster global hub in Red Hat Advanced Cluster Management. For more details, see Hub cluster network configuration .

1.3.1.1. Installing multicluster global hub by using the console

To install the multicluster global hub operator in a connected environment by using the OpenShift Container Platform console, complete the following steps:

1. Log in to the OpenShift Container Platform console as a user with the **cluster-admin** role.
2. From the navigation menu, select **Operators** > the **OperatorHub** icon.

3. Locate and select the **Multicluster global hub operator**.
4. Click **Install** to start the installation.
5. After the installation completes, check the status on the *Installed Operators* page.
6. Click **Multicluster global hub operator** to go to the *Operator* page.
7. Click the **Multicluster global hub** tab to see the **Multicluster Global Hub** instance.
8. Click **Create Multicluster Global Hub** to create the **Multicluster Global Hub** instance.
9. Enter the required information and click **Create** to create the **Multicluster Global Hub** instance.

Notes:

- The multicluster global hub is only available for the x86 platform.
- The policy and application are disabled in Red Hat Advanced Cluster Management after the multicluster global hub is installed.

1.3.2. Additional resources

- For more information about mirroring an Operator catalog, see Mirroring an Operator catalog .
- For more information about accessing images from private registries, see Accessing images for Operators from private registries.
- For more information about adding a catalog source, see Adding a catalog source to a cluster .
- For more information about installing Red Hat Advanced Cluster Management in a disconnected environment, see Install in disconnected network environments .
- For more information about mirroring images, see Mirroring images for a disconnected installation.
- For more information about the Operator SDK Intregration with OLM, see Operator SDK Integration with Operator Lifecycle Manager.

1.4. INSTALLING MULTICLUSTER GLOBAL HUB IN A DISCONNECTED ENVIRONMENT

If your cluster is in a restricted network, you can deploy the multicluster global hub operator in the disconnected environment.

Required access: Cluster administrator

1.4.1. Prerequisites

You must meet the following requirements before you install multicluster global hub in a disconnected environment:

- An image registry and a bastion host must have access to both the internet and to your mirror registry.

- Install the Operator Lifecycle Manager on your cluster. See Operator Lifecycle Manager (OLM).
- Install Red Hat Advanced Cluster Management for Kubernetes.
- Install the following command line interfaces:
  - The OpenShift Container Platform command line. See Getting started with the OpenShift Container Platform CLI.
  - The **opm** command line. See Installing the opm CLI.
  - The **oc-mirror** plugin. See, Mirroring images for a disconnected installation using the oc-plugin.

1.4.2. Configuring a mirror registry

Installing multicluster global hub in a disconnected environment involves the use of a local mirror image registry. At this point, it is assumed that you have set up a mirror registry during the OpenShift Container Platform cluster installation.

Complete the following procedures to provision the mirror registry for multicluster global hub:

1.4.2.1. Creating operator packages in mirror catalog with oc-mirror plug-in

Red Hat provides the multicluster global hub and AMQ Streams operators in the Red Hat operators catalog, which are delivered by the **registry.redhat.io/redhat/redhat-operator-index** index image. When you prepare your mirror of this catalog index image, you can choose to either mirror the entire catalog as provided by Red Hat, or you can mirror a subset that contains only the operator packages that you intend to use.

If you are creating a full mirror catalog, no special considerations are needed as all of the packages required to install multicluster global hub and AMQ Streams are included. However, if you are creating a partial or filtered mirrored catalog, for which you identify particular packages to be included, you must to include the **multicluster-global-hub-operator-rh** and **amq-streams** package names in your list.

Complete the following steps to create a local mirror registry of the **multicluster-global-hub-operator-rh** and **amq-streams** packages:

1. Create a **ImageSetConfiguration** YAML file to configure and add the operator image. Your YAML file might resemble the following content, with the current version replacing **4.x**:

```
kind: ImageSetConfiguration
apiVersion: mirror.openshift.io/v1alpha2
storageConfig:
  registry:
     imageURL: myregistry.example.com:5000/mirror/oc-mirror-metadata
mirror:
  platform:
    channels:
    - name: stable-4.x
      type: ocp
  operators:
  - catalog: registry.redhat.io/redhat/redhat-operator-index:v4.12
    packages:
    - name: multicluster-global-hub-operator-rh
```

```
    - name: amq-streams
  additionalImages: []
  helm: {}
```

2. Mirror the image set directly to the target mirror registry by using the following command:

```
oc mirror --config=./imageset-config.yaml docker://myregistry.example.com:5000
```

3. Mirror the image set in a fully disconnected environment. For more details, see Mirroring images for a disconnected installation.

1.4.2.2. Adding the registry and catalog to your disconnected cluster

To make your mirror registry and catalog available on your disconnected cluster. Complete the following steps:

1. Disable the default catalog sources of Operator Hub. Run the following command to update the **OperatorHub** resource:

```
oc patch OperatorHub cluster --type json \
-p '[{"op": "add", "path": "/spec/disableAllDefaultSources", "value": true}]'
```

2. Mirror the Operator catalog by completing the procedure, Mirroring the Operator catalog.

3. Add the **CatalogSource** resource for your mirrored catalog into the **openshift-marketplace** namespace. Your **CatalogSource** YAML file might be similar to the following example:

```
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: my-mirror-catalog-source
  namespace: openshift-marketplace
spec:
  image: myregistry.example.com:5000/mirror/my-operator-index:v4.12
  sourceType: grpc
  secrets:
  - <global-hub-secret>
```

- **Note:** Take note of the value of the **metadata.name** field.

4. Save the updated file.

5. Verify that the required packages are available from your disconnected cluster by querying the available **PackageManifest** resources. Run the following command: with the following command:

```
oc -n openshift-marketplace get packagemanifests
```

The list that is displayed should include entries showing that the **multicluster-global-hub-operator-rh** and **amq-streams** packages are supplied by the catalog source for your mirror catalog:

1.4.3. Configuring the image registry

In order to have your cluster obtain container images for the multicluster global hub operator from your local mirror registry, rather than from the internet-hosted registries, you must configure an **ImageContentSourcePolicy** resource on your disconnected cluster to redirect image references to your mirror registry. The **ImageContentSourcePolicy** only support the image mirror with image **digest**.

If you mirrored your catalog using the **oc adm catalog mirror** command, the needed image content source policy configuration is in the **imageContentSourcePolicy.yaml** file inside of the **manifests-*** directory that is created by that command.

If you used the **oc-mirror** plug-in to mirror your catalog instead, the **imageContentSourcePolicy.yaml** file is within the **oc-mirror-workspace/results-*** directory create by the oc-mirror plug-in.

In either case, you can apply the policies to your disconnected command using an **oc apply** or **oc replace** command such as **oc replace -f ./<path>/imageContentSourcePolicy.yaml**

The required image content source policy statements can vary based on how you created your mirror registry, but are similar to this example:

```
apiVersion: operator.openshift.io/v1alpha1
kind: ImageContentSourcePolicy
metadata:
  labels:
    operators.openshift.org/catalog: "true"
  name: global-hub-operator-icsp
spec:
  repositoryDigestMirrors:
  - mirrors:
    - myregistry.example.com:5000/multicluster-globalhub
    source: registry.redhat.io/multicluster-globalhub
  - mirrors:
    - myregistry.example.com:5000/openshift4
    source: registry.redhat.io/openshift4
  - mirrors:
    - myregistry.example.com:5000/redhat
    source: registry.redhat.io/redhat
```

You can configure different image registries for different managed hubs with the **ManagedClusterImageRegistry**. See Importing a cluster that has a ManagedClusterImageRegistry to use the **ManagedClusterImageRegistry** API to replace the agent image.

By completing the previous step, a label and an annotation are added to the selected **ManagedCluster**. This means that the agent image in the cluster are replaced with the mirror image.

- Label: **multicluster-global-hub.io/image-registry= <namespace.managedclusterimageregistry-name>**
- Annotation: **multicluster-global-hub.io/image-registries: <image-registry-info>**

1.4.3.1. Configure the image pull secret

If the Operator or Operand images that are referenced by a subscribed Operator require access to a private registry, you can either provide access to all namespaces in the cluster, or to individual target tenant namespaces.

1.4.3.1.1. Configure the multicluster global hub image pull secret in an OpenShift Container Platform cluster

You can configure the image pull secret in an existing OpenShift Container Platform cluster.

Note: Applying the image pull secret on a pre-existing cluster causes a rolling restart of all of the nodes.

Complete the following steps to configure the pull secret:

1. Export the user name from the pull secret:

```
export USER=<the-registry-user>
```

2. Export the password from the pull secret:

```
export PASSWORD=<the-registry-password>
```

3. Copy the pull secret:

```
oc get secret/pull-secret -n openshift-config --template='{{index .data ".dockerconfigjson" | base64decode}}' > pull_secret.yaml
```

4. Log in using the pull secret:

```
oc registry login --registry=${REGISTRY} --auth-basic="$USER:$PASSWORD" --to=pull_secret.yaml
```

5. Specify the multicluster global hub image pull secret:

```
oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=pull_secret.yaml
```

6. Remove the old pull secret:

```
rm pull_secret.yaml
```

1.4.3.1.2. Configure the multicluster global hub image pull secret to an individual namespace

You can configure the image pull secret to an individual namespace by completing the following steps:

1. Create the secret in the tenant namespace by running the following command:

```
oc create secret generic <secret_name> -n <tenant_namespace> \
--from-file=.dockerconfigjson=<path/to/registry/credentials> \
--type=kubernetes.io/dockerconfigjson
```

2. Link the secret to the service account for your operator or operand:

```
oc secrets link <operator_sa> -n <tenant_namespace> <secret_name> --for=pull
```

1.4.3.2. Installing the Global Hub Operator

You can install and subscribe an Operator from OperatorHub using the Red Hat OpenShift Container Platform web console. See Adding Operators to a cluster for the procedure. After adding the Operator, you can check the status of the multicluster global hub Operator by running the following command:

```
oc get pods -n multicluster-global-hub
NAME                                         READY   STATUS    RESTARTS   AGE
multicluster-global-hub-operator-687584cb7c-fnftj   1/1     Running   0          2m12s
```

1.4.4. Additional resources

- For more information about creating a mirror registry, see Create a mirror registry.
- For more information about mirroring images, see Mirroring images for a disconnected installation.
- For more information about mirroring an Operator catalog, see Mirroring an Operator catalog.

1.5. INTEGRATING EXISTING COMPONENTS

The multicluster global hub requires middleware components, Kafka and PostgreSQL, along with Grafana as the Observability platform to provide the policy compliance view. The multicluster global hub provides versions of Kafka, PostgreSQL, and Grafana. You can also integrate your own existing Kafka, PostgreSQL, and Grafana.

- Integrating an existing version of Kafka
- Integrating an existing version of PostgreSQL
- Integrating an existing version of Grafana

1.5.1. Integrating an existing version of Kafka

If you have your own instance of Kafka, you can use it as the transport for multicluster global hub. Kafka 3.3 is the tested version. Complete the following steps to integrate an instance of Kafka:

1. If you do not have a persistent volume for your Kafka instance, you need to create one.
2. Create a secret named **multicluster-global-hub-transport** in the **multicluster-global-hub** namespace.
    a. Extract the information in the following required fields:
        - **bootstrap.servers**: Specifies the Kafka bootstrap servers.
        - **ca.crt**: Required if you use the **KafkaUser** custom resource to configure authentication credentials. See the *User authentication* topic in the STRIMZI documentation for the required steps to extract the **ca.crt** certificate from the secret.
        - **client.crt**: Required, see the *User authentication* topic in the STRIMZI documentation for the steps to extract the **user.crt** certificate from the secret.
        - **client.key**: Required, see the *User authentication* topic in the STRIMZI documentation for the steps to extract the **user.key** from the secret.
3. Create the secret by running the following command, replacing the values with your extracted values where necessary:

```
oc create secret generic multicluster-global-hub-transport -n multicluster-global-hub \
    --from-literal=bootstrap_server=<kafka-bootstrap-server-address> \
```

```
    --from-file=ca.crt=<CA-cert-for-kafka-server> \
    --from-file=client.crt=<Client-cert-for-kafka-server> \
    --from-file=client.key=<Client-key-for-kafka-server>
```

4. If automatic topic creation is configured on your Kafka instance, then skip this step. If it is not configured, create the **spec**, **status**, and **event** topics manually.

5. Ensure that the global hub user that accesses Kafka has the permission to read data from the topics and write data to the topics.

1.5.2. Integrating an existing version of PostgreSQL

If you have your own PostgreSQL relational database, you can use it as the storage for multicluster global hub. PostgreSQL 13 is the tested version.

The minimum required storage size is 20GB. This amount can store 3 managed hubs with 250 managed clusters and 50 policies per managed hub for 18 months. You need to create a secret named **multicluster-global-hub-storage** in the **multicluster-global-hub** namespace. The secret must contain the following fields:

- **database_uri**: It is used to create the database and insert data. Your value must resemble the following format: **postgres://<user>:<password>@<host>:<port>/<database>?sslmode=<mode>**.
- **database_uri_with_readonlyuser**: It is used to query data by the instance of Grafana that is used by multicluster global hub. It is an optional value. Your value must resemble the following format: **postgres://<user>:<password>@<host>:<port>/<database>?sslmode=<mode>**.
- The **ca.crt**, which is based on the **sslmode**, is an optional value.

    1. Verify that your cluster has the minimum required storage size of 20GB. This amount can store three managed hubs with 250 managed clusters and 50 policies per managed hub for 18 months.

    2. Create the secret by running the following command:

```
oc create secret generic multicluster-global-hub-storage -n multicluster-global-hub \
  --from-literal=database_uri=<postgresql-uri> \
  --from-literal=database_uri_with_readonlyuser=<postgresql-uri-with-readonlyuser> \
  --from-file=ca.crt=<CA-for-postgres-server>
```

The host must be accessible from the multicluster global hub cluster. If your PostgreSQL database is in a Kubernetes cluster, you can consider using the service type with **nodePort** or **LoadBalancer** to expose the database. For more information, see Accessing the provisioned postgres database for troubleshooting.

1.5.3. Integrating an existing version of Grafana

Using an existing Grafana instance might work with multicluster global hub if you are relying on your own Grafana to get metrics from multiple sources, such as Prometheus, from different clusters and if you aggregate the metrics yourself. To get multicluster global hub data into your own Grafana, you need to configure the data source and import the dashboards.

1. Collect the PostgreSQL connection information from the multicluster global hub Grafana **datasource** secret by running the following command:

```
oc get secret multicluster-global-hub-grafana-datasources -n multicluster-global-hub -
ojsonpath='{.data.datasources\.yaml}' | base64 -d
```

The output resembles the following example:

```
apiVersion: 1
datasources:
- access: proxy
  isDefault: true
  name: Global-Hub-DataSource
  type: postgres
  url: postgres-primary.multicluster-global-hub.svc:5432
  database: hoh
  user: guest
  jsonData:
    sslmode: verify-ca
    tlsAuth: true
    tlsAuthWithCACert: true
    tlsConfigurationMethod: file-content
    tlsSkipVerify: true
    queryTimeout: 300s
    timeInterval: 30s
  secureJsonData:
    password: xxxxx
    tlsCACert: xxxxx
```

2. Configure the **datasource** in your own Grafana instance by adding a source, such as PostgreSQL, and complete the required fields with the information you previously extracted. See the following required fields:

   - Name
   - Host
   - Database
   - User
   - Password
   - TLS/SSL Mode
   - TLS/SSL Method
   - CA Cert

3. If your Grafana is not in the multicluster global hub cluster, you need to expose the PostgreSQL by using the **LoadBalancer** so the PostgreSQL can be accessed from outside. You can add the following value into the **PostgresCluster** operand:

```
service:
  type: LoadBalancer
```

After you add that content, then you can get the **EXTERNAL-IP** from the **postgres-ha** service. See the following example:

```
oc get svc postgres-ha -n multicluster-global-hub
NAME          TYPE           CLUSTER-IP     EXTERNAL-IP                          PORT(S)          AGE
postgres-ha   LoadBalancer   172.30.227.58  xxxx.us-east-1.elb.amazonaws.com     5432:31442/TCP   128m
```

After running that command, you can use **xxxx.us-east-1.elb.amazonaws.com:5432** as the PostgreSQL Connection Host.

4. Import the existing dashboards.

   a. Follow the steps in Export and import dashboards in the official Grafana documentation to export the dashboard from the existing Grafana instance.

   b. Follow the steps in Export and import dashboards in the official Grafana documentation to import a dashboard into the multicluster global hub Grafana instance.

1.5.4. Additional resources

See User authentication in the STRIMZI documentation for more information about how to extract the **ca.crt** certificate from the secret.

See User authentication in the STRIMZI documentation for the steps to extract the **user.crt** certificate from the secret.

1.6. IMPORTING A MANAGED HUB CLUSTER IN THE DEFAULT MODE

To import an existing hub cluster as a managed hub cluster, complete the following steps:

1. Disable the cluster self-management in the existing Red Hat Advanced Cluster Management for Kubernetes hub cluster by setting the **disableHubSelfManagement** setting to **true** in the **multiclusterhub** custom resource. This setting disables the automatic importing of the hub cluster as a managed cluster.

2. Import the managed hub cluster by completing the steps in Cluster import introduction.

3. After the managed hub cluster is imported, check the multicluster global hub agent status to ensure that the agent is running in the managed hub cluster by running the following command:

```
oc get managedclusteraddon multicluster-global-hub-controller -n
$<managed_hub_cluster_name>
```

1.7. ACCESSING THE GRAFANA DATA

The Grafana data is exposed through the route. Run the following command to display the login URL:

```
oc get route multicluster-global-hub-grafana -n <the-namespace-of-multicluster-global-hub-instance>
```

The authentication method of this URL is same as authenticating to the Red Hat OpenShift Container Platform console.

1.7.1. Viewing policy status by using Grafana dashboards

After accessing the global hub Grafana data, you can monitor the policies that were configured through the hub cluster environments that are managed.

From the multicluster global hub dashboard, you can identify the compliance status of the policies of the system over a selected time range. The policy compliance status is updated daily, so the dashboard does not display the status of the current day until the following day.

From the global hub dashboard, you can identify the compliance status of the policies of the system over a selected time range. The policy compliance status is updated daily, so the dashboard does not display the status of the current day until the following day.

To navigate the multicluster global hub dashboards, you can observe and filter the policy data by grouping them by **policy** or by **cluster**.

If you prefer to examine the policy data by using the **policy** grouping, start from the and the dashboard called **Global Hub - Policy Group Compliancy Overview**.

This dashboard allows you to filter the policy data based on **standard**, **category**, and **control**. After selecting a specific point in time on the graph, you are directed to the **Global Hub - Offending Policies** dashboard. The **Global Hub - Offending Policies** dashboard lists the non-compliant or unknown policies at that time. After selecting a target policy, you can view related events and see what has changed by accessing the **Global Hub - What's Changed** / **Policies** dashboard.

Similarly, if you want to examine the policy data by **cluster** grouping, begin by using the **Global Hub - Cluster Group Compliancy Overview** dashboard. The navigation flow is identical to the **policy** grouping flow, but you select filters that are related to the cluster, such as managed cluster **labels** and **values**. Instead of viewing policy events for all clusters, after reaching the **Global Hub - What's Changed** / **Clusters** dashboard, you can view policy events related to an individual cluster.

1.8. GRAFANA ALERTS (TECHNOLOGY PREVIEW)

You can configure three Grafana alerts, which are stored in the **multicluster-global-hub-default-alerting** config map. These alerts notify you of suspicious policies, suspicious clusters compliance status change, and failed cron jobs.

See the following descriptions of the alerts:

- Suspicious policy change: This alert rule watches the suspicious policies change. If the following events occur more than five times in one hour, it creates notifications.
    - A policy was enabled or disabled.
    - A policy was updated.
- Suspicious cluster compliance status change: This alert rule watches the cluster compliance status and policy events for a cluster. There are two rules in this alert:
    - Cluster compliance status changes frequently: If a cluster compliance status changes from **compliance** to **non-compliance** more than three times in one hour, it creates notifications.
    - Too many policy events in a cluster: For a policy in a cluster, if there are more than 20 events in five minutes, it creates notifications. If this alert is always firing, the data in the **event.local_policies** table increases too fast.
- Cron Job failed: This alert watches the cron jobs that are described in Configuring the cron jobs for failed events. There are two rules in this alert:
    - Local compliance job failed: If this alert rule creates notifications, it means the local compliance status synchronization job failed. It might cause the data in the **history.local_compliance** table to be lost. Run the job manually, if necessary.

- Data retention job failed: If this alert rule starts creating notifications, it means the data retention job failed. You can run it manually.

1.8.1. Deleting a default Grafana alert rule

If the default Grafana alert rules do not provide useful information, you can delete the Grafana alert rule by including a **deleteRules** section in the **multicluster-global-hub-custom-alerting** config map. See Customize Grafana alerting resources for more information about the **multicluster-global-hub-custom-alerting** config map.

To delete all of the default alerts, the **deleteRules** configuration section should resemble the following example:

```
deleteRules:
  - orgId: 1
    uid: globalhub_suspicious_policy_change
  - orgId: 1
    uid: globalhub_cluster_compliance_status_change_frequently
  - orgId: 1
    uid: globalhub_high_number_of_policy_events
  - orgId: 1
    uid: globalhub_data_retention_job
  - orgId: 1
    uid: globalhub_local_compliance_job
```

1.8.2. Customizing Grafana alerts

The multicluster global hub supports creating custom Grafana alerts. Complete the following steps to customize your Grafana alerts:

1.8.2.1. Customizing your grafana.ini file

To customize your **grafana.ini** file, create a secret named **multicluster-global-hub-custom-grafana-config** in the namespace where you installed your multicluster global hub operator. The secret data key is **grafana.ini**, as seen in the following example. Replace the required information with your own credentials:

```
apiVersion: v1
kind: Secret
metadata:
  name: multicluster-global-hub-custom-grafana-config
  namespace: multicluster-global-hub
type: Opaque
stringData:
  grafana.ini: |
    [smtp]
    enabled = true
    host = smtp.google.com:465
    user = <example@google.com>
    password = <password>
    ;cert_file =
    ;key_file =
    skip_verify = true
```

```
from_address = <example@google.com>
from_name = Grafana
;ehlo_identity = dashboard.example.com 1
```

<1>The **EHLO** identity in the **SMTP** dialog, which defaults to **instance_name**.

Note: You cannot configure the section that already contains the **multicluster-global-hub-default-grafana-config** secret.

1.8.2.2. Customizing Grafana alerting resources

The multicluster global hub supports customizing the alerting resources, which is explained in Create and manage alerting resources using file provisioning in the Grafana documentation.

To customize the alerting resources, create a config map named **multicluster-global-hub-custom-alerting** in the **multicluster-global-hub** namespace.

The config map data key is **alerting.yaml**, as in the following example:

```
apiVersion: v1
data:
  alerting.yaml: |
    contactPoints:
      - orgId: 1
        name: globalhub_policy
        receivers:
          - uid: globalhub_policy_alert_email
            type: email
            settings:
              addresses: <example@redhat.com>
              singleEmail: false
          - uid: globalhub_policy_alert_slack
            type: slack
            settings:
              url: <Slack-webhook-URL>
              title: |
                {{ template "globalhub.policy.title" . }}
              text: |
                {{ template "globalhub.policy.message" . }}
    policies:
      - orgId: 1
        receiver: globalhub_policy
        group_by: ['grafana_folder', 'alertname']
        matchers:
          - grafana_folder = Policy
        repeat_interval: 1d
    deleteRules:
      - orgId: 1
        uid: [Alert Rule Uid]
    muteTimes:
      - orgId: 1
        name: mti_1
        time_intervals:
          - times:
              - start_time: '06:00'
                end_time: '23:59'
```

```
        location: 'UTC'
      weekdays: ['monday:wednesday', 'saturday', 'sunday']
      months: ['1:3', 'may:august', 'december']
      years: ['2020:2022', '2030']
      days_of_month: ['1:5', '-3:-1']
kind: ConfigMap
metadata:
  name: multicluster-global-hub-custom-alerting
  namespace: multicluster-global-hub
```

1.9. CONFIGURING THE CRON JOBS

You can configure the cron job settings of the multicluster global hub.

After installing the multicluster global hub operand, the multicluster global hub manager runs and displays a job scheduler for you to schedule the following cron jobs:

- Local compliance status sync job: This cron job runs at midnight every day, based on the policy status and events collected by the manager on the previous day. Running this job summarizes the compliance status and the change frequency of the policy on the cluster, and stores them to the **history.local_compliance** table as the data source of the Grafana dashboards.

- Data retention job: Some data tables in multicluster global hub continue to grow over time, which normally can cause problems when the tables get too large. The following two methods help to minimize the issues that result from tables that are too large:

  - Deleting older data that is no longer needed

  - Enabling partitioning on the large table to run queries and deletions on faster
    For event tables like the **event.local_policies** and the **history.local_compliance** that increase in size daily, range partitioning divides the large tables into smaller partitions. This process also creates the partition tables for the next month each time it is run. For the policy and cluster tables like **local_spec.policies** and **status.managed_clusters**, there are **deleted_at** indexes on the tables to improve performance when hard deleting.

    You can change the duration of time that the data is retained by changing the **retention** setting on the multicluster global hub operand. The recommended minimum value is 1 month, and the default value is 18 months. The run interval of this job should be less than one month.

The listed cron jobs run every time the multicluster global hub manager starts. The local compliance status sync job is run once a day and can be run multiple times within the day without changing the result. The data retention job is run once a week and also can be run many times per month without a change in the results.

The status of these jobs are are saved in the metrics named **multicluster_global_hub_jobs_status**, which can be viewed from the console of the Red Hat OpenShift Container Platform cluster. A value of **0** indicates that the job ran successfully, while a value of **1** indicates failure.

If there is a failed job, you can troubleshoot by using the log tables (**history.local_compliance_job_log**, **event.data_retention_job_log**). See Restoring compliance data for more details and for guidance for deciding whether to run the service manually.

1.10. RUNNING THE SUMMARIZATION PROCESS MANUALLY

You can also run the summarization process manually. This can be helpful when you are trying to investigate a problem or need a report sooner than the next scheduled routine.

The manual summarization process consists of two subtasks:

- Insert the cluster policy data of that day from Materialized View **local_compliance_view_<yyyy_MM_dd>** to **history.local_compliance**.
- Update the **compliance** and policy flip **frequency** of that day to **history.local_compliance** based on **event.local_policies**.

Complete the following steps to run the summarization process manually:

1. Connect to the database.
   You can use clients such as pgAdmin, tablePlush, and so on, to connect to the multicluster global hub database to run the SQL statements in the next few steps. You can directly connect to the database on the cluster by running the following command:

   ```
   oc exec -it multicluster-global-hub-postgres-0 -n multicluster-global-hub -- psql -d hoh
   ```

2. Determine the date when it needs to be run, such as **2023-07-06**.
   If you find that there is no compliance information on the dashboard for **2023-07-06**, then find the job failure information of the day following this day in the **history.local_compliance_job_log**. In this case, it is **2023-07-07**. It can be determined that **2023-07-06** is the date when we need to manually run the summary processes.

3. Check whether the Materialized View of **history.local_compliance_view_2023_07_06** exists by running the following command:

   ```
   select * from history.local_compliance_view_2023_07_06;
   ```

   If the view exists, load the view records to **history.local_compliance** by runnning the following command:

   ```
   -- exec the insert func for that day '2023_07_06'
   SELECT history.insert_local_compliance_job('2023_07_06');
   ```

   If the view does not exist, inherit the history compliance records of the day before that day, in this example, it might be **2023_07_05**.

   ```
   -- call the func to generate the data of '2023_07_06' by inheriting '2023_07_05'
   CALL history.inherit_local_compliance_job('2023_07_05', '2023_07_06');
   ```

4. Update the **compliance** and **frequency** information of that day to **history.local_compliance**.

   ```
   -- call the func to update records start with '2023-07-06', end with '2023-07-07'
   SELECT history.update_local_compliance_job('2023_07_06', '2023_07_07');
   ```

5. Find the records of that day generated in **history.local_compliance**. You can safely delete the Materialized View **history.local_compliance_view_2023_07_06** by running the following command:

   ```
   DROP MATERIALIZED VIEW IF EXISTS history.local_compliance_view_2023_07_06;
   ```