Red Hat Advanced Cluster Management for Kubernetes 2.12

Networking

Networking

Last Updated: 2024-11-05

Red Hat Advanced Cluster Management for Kubernetes 2.12 Networking

Networking

Legal Notice

Abstract

Read more to learn about networking requirements for the hub cluster and managed cluster.

Table of Contents

CHAPTER 1. NETWORKING

Learn about network requirements for both the hub cluster and the managed cluster.

- Hub cluster network configuration
- Managed cluster network configuration
- Advanced network configuration
- Submariner multicluster networking and service discovery

1.1. HUB CLUSTER NETWORK CONFIGURATION

Important: The trusted CA bundle is available in the Red Hat Advanced Cluster Management 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.

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

1.1.1. Hub cluster network configuration table

See the hub cluster network requirements in the following table:

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Outbound to the managed cluster	HTTPS	Retrieval of logs dynamically from Search console for the pods of the managed cluster, uses the **klusterlet-addon-workmgr** service that is running on the managed cluster	443	None	IP address to access managed cluster route

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Outbound to the managed cluster	HTTPS	Kubernetes API server of the managed cluster that is provisioned during installation to install the klusterlet	6443	None	IP of Kubernetes managed cluster API server
Outbound to the channel source	HTTPS	The channel source, including GitHub, Object Store, and Helm repository, which is only required when you are using Application lifecycle, OpenShift GitOps, or Argo CD to connect	443	None	IP of the channel source
Inbound from the managed cluster	HTTPS	Managed cluster to push metrics and alerts that are gathered only for managed clusters that are running on a supported OpenShift Container Platform version	443	None	IP address to hub cluster access route
Inbound from the managed cluster	HTTPS	Kubernetes API Server of hub cluster that is watched for changes from the managed cluster	6443	None	IP address of hub cluster Kubernetes API Server

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Outbound to the ObjectStore	HTTPS	Sends Observability metric data for long term storage when the Cluster Backup Operator is running	443	None	IP address of ObjectStore
Outbound to the image repository	HTTPS	Access images for OpenShift Container Platform and Red Hat Advanced Cluster Management	443	None	IP address of image repository

1.2. MANAGED CLUSTER NETWORK CONFIGURATION

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

1.2.1. Managed cluster network configuration table

See the managed cluster network requirements in the following table:

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Inbound from the hub cluster	HTTPS	Sending of logs dynamically from Search console for the pods of the managed cluster, uses the **klusterlet-addon-workmgr** service that is running on the managed cluster	443	None	IP address to access managed cluster route

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Inbound from the hub cluster	HTTPS	Kubernetes API server of the managed cluster that is provisioned during installation to install the klusterlet	6443	None	IP of Kubernetes managed cluster API server
Outbound to the image repository	HTTPS	Access images for OpenShift Container Platform and Red Hat Advanced Cluster Management	443	None	IP address of image repository
Outbound to the hub cluster	HTTPS	Managed cluster to push metrics and alerts that are gathered only for managed clusters that are running on a supported OpenShift Container Platform version	443	None	IP address to hub cluster access route
Outbound to the hub cluster	HTTPS	Watches the Kubernetes API server of the hub cluster for changes	6443	None	IP address of hub cluster Kubernetes API Server

Direction	Protocol	Connection	Port (if specified)	Source address	Destination address
Outbound to the channel source	HTTPS	The channel source, including GitHub, Object Store, and Helm repository, which is only required when you are using Application lifecycle, OpenShift GitOps, or Argo CD to connect	443	None	IP of the channel source

1.3. ADVANCED NETWORK CONFIGURATION

- Additional networking requirements for infrastructure operator table
- Submariner networking requirements table
- Additional networking requirements for Hive table
- Application deployment network requirements table
- Namespace connection network requirements table

1.3.1. Additional networking requirements for infrastructure operator table

When you are installing bare metal managed clusters with the Infrastructure Operator, see Network configuration in the multicluster engine for Kubernetes operator documentation for additional networking requirements.

1.3.2. Submariner networking requirements table

Clusters that are using Submariner require three open ports. The following table shows which ports you might use:

Direction	Protocol	Connection	Port (if specified)
Outbound and inbound	UDP	Each of the managed clusters	4800

Direction	Protocol	Connection	Port (if specified)
Outbound and inbound	UDP	Each of the managed clusters	4500, 500, and any other ports that are used for IPSec traffic on the gateway nodes
Inbound	TCP	Each of the managed clusters	8080

1.3.3. Additional networking requirements for Hive table

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.

1.3.4. Application deployment network requirements table

In general, the application deployment communication is one way from a managed cluster to the hub cluster. The connection uses **kubeconfig**, which is configured by the agent on the managed cluster. The application deployment on the managed cluster needs to access the following namespaces on the hub cluster:

- The namespace of the channel resource
- The namespace of the managed cluster

1.3.5. Namespace connection network requirements table

- Application lifecycle connections:
  - The namespace **open-cluster-management** needs to access the console API on port 4000.
  - The namespace **open-cluster-management** needs to expose the Application UI on port 3001.
- Application lifecycle backend components (pods):

On the hub cluster, all of the application lifecycle pods are installed in the **open-cluster-management** namespace, including the following pods:

- multicluster-operators-hub-subscription
- multicluster-operators-standalone-subscription
- multicluster-operators-channel
- multicluster-operators-application
- multicluster-integrations
  As a result of these pods being in the **open-cluster-management** namespace:
- The namespace **open-cluster-management** needs to access the Kube API on port 6443.

On the managed cluster, only the **klusterlet-addon-appmgr** application lifecycle pod is installed in the **open-cluster-management-agent-addon** namespace:

- The namespace **open-cluster-management-agent-addon** needs to access the Kube API on port 6443.

- Governance and risk:
  On the hub cluster, the following access is required:
  - The namespace **open-cluster-management** needs to access the Kube API on port 6443.
  - The namespace **open-cluster-management** needs to access the OpenShift DNS on port 5353.

  On the managed cluster, the following access is required:
  - The namespace **open-cluster-management-addon** needs to access the Kube API on port 6443.

1.4. SUBMARINER MULTICLUSTER NETWORKING AND SERVICE DISCOVERY

Submariner is an open source tool that you can use with Red Hat Advanced Cluster Management for Kubernetes to provide direct networking and service discovery between two or more managed clusters in your environment, either on-premises or in the cloud. Submariner is compatible with Multi-Cluster Services API (Kubernetes Enhancements Proposal #1645). For more information about Submariner, see the Submariner site .

Submariner is not supported with all of the infrastructure providers that Red Hat Advanced Cluster Management can manage. Refer to the Red Hat Advanced Cluster Management support matrix for more details about the support levels of infrastructure providers, including which providers support automated console deployments or require manual deployment.

See the following topics to learn more about how to use Submariner:

- Deploying Submariner on disconnected clusters
- Configuring Submariner
- Installing the subctl command utility

- Deploying Submariner by using the console
- Deploying Submariner manually
- Customizing Submariner deployments
- Managing service discovery for Submariner
- Uninstalling Submariner

1.4.1. Deploying Submariner on disconnected clusters

Deploying Submariner on disconnected clusters can help with security concerns by reducing the risk of external attacks on clusters. To deploy Submariner with Red Hat Advanced Cluster Management for Kubernetes on disconnected clusters, you must first complete the steps outlined in Install in disconnected network environments.

1.4.1.1. Configuring Submariner on disconnected clusters

After following the steps outlined in Install in disconnected network environments , you must configure Submariner during the installation to support deployment on disconnected clusters. See the following topics:

1.4.1.1.1. Mirroring images in the local registry

Make sure to mirror the **Submariner Operator bundle** image in the local registry before deploying Submariner on disconnected clusters.

1.4.1.1.2. Customizing *catalogSource* names

By default, **submariner-addon** searches for a **catalogSource** with the name **redhat-operators**. When using a **catalogSource** with a different name, you must update the value of the **SubmarinerConfig.Spec.subscriptionConfig.Source** parameter in the **SubmarinerConfig** associated with your managed cluster with the custom name of the **catalogSource**.

1.4.1.1.3. Enabling *airGappedDeployment* in *SubmarinerConfig*

When installing **submariner-addon** on a managed cluster from the Red Hat Advanced Cluster Management for Kubernetes console, you can select the **Disconnected cluster** option so that Submariner does not make API queries to external servers.

If you are installing Submariner by using the APIs, you must set the **airGappedDeployment** parameter to **true** in the **SubmarinerConfig** associated with your managed cluster.

1.4.2. Configuring Submariner

Red Hat Advanced Cluster Management for Kubernetes provides Submariner as an add-on for your hub cluster. To learn how to configure Submariner, read the following topics:

- Prerequisites
- Submariner ports table
- Globalnet

1.4.2.1. Prerequisites

Ensure that you have the following prerequisites before using Submariner:

- A credential to access the hub cluster with **cluster-admin** permissions.
- IP connectivity must be configured between the gateway nodes. When connecting two clusters, at least one of the clusters must be accessible to the gateway node by using its public or private IP address designated to the gateway node. See *Submariner NAT Traversal* for more information.
- If you are using OVN Kubernetes, clusters must be on a supported version of OpenShift Container Platform.
- If your OpenShift Container Platform clusters use OpenShift SDN CNI, the firewall configuration across all nodes in each of the managed clusters must allow 4800/UDP in both directions.
- The firewall configuration must allow 4500/UDP and 4490/UDP on the gateway nodes for establishing tunnels between the managed clusters.
- For OpenShift Container Platform ARM deployments, you must use the **c6g.large instanceType** or any other available instance type.
- If the gateway nodes are directly reachable over their private IPs without any NAT in between, make sure that the firewall configuration allows the ESP protocol on the gateway nodes. **Note:** This is configured automatically when your clusters are deployed in an Amazon Web Services, Google Cloud Platform, Microsoft Azure, or Red Hat OpenStack environment, but must be configured manually for clusters on other environments and for the firewalls that protect private clouds.
- The **managedcluster** name must follow the DNS label standard as defined in RFC 1123 and meet the following requirements:
  - Contain 63 characters or fewer
  - Contain only lowercase alphanumeric characters or '-'
  - Start with an alphanumeric character
  - End with an alphanumeric character

1.4.2.2. Submariner ports table

View the following table to see the Submariner ports that you must enable:

Name	Default value	Customizable	Optional or required
IPsec NATT	4500/UDP	Yes	Required
VXLAN	4800/UDP	No	Required
NAT discovery port	4490/UDP	No	Required

1.4.2.3. Configuring Submariner for an existing VPC

If you installed your cluster into an existing Amazon Virtual Private Cloud (VPC) on Amazon Web Services (AWS), you must complete the following steps to configure Subarminer:

1. Open and edit the **SubmarinerConfig** file by running the following command. Replace values where needed:

```
oc edit submarinerconfig -n <managed-cluster-ns> submariner
```

2. Add the following annotations in the **metadata** field. Replace values where needed:
Note: All the IDs you need to add are alphanumeric.

```
annotations:
  submariner.io/control-plane-sg-id: <control-plane-group-id> 1
  submariner.io/subnet-id-list: <subnet-id-list> 2
  submariner.io/vpc-id: <custom-vpc-id> 3
  submariner.io/worker-sg-id: <worker-security-group-id> 4
```

1. Replace with your control plane security group ID. Typically, you can find the ID from the control plane security group that has a name similar to **<infra-id>-master-sg>**.
2. Replace with a comma-separated list of public subnet IDs in your custom VPC.
3. Replace with your custom VPC ID.
4. Replace with your worker security group ID. Typically, you can find the ID from the worker security group that has a name similar to **<infra-id>-worker-sg**.

1.4.2.4. Globalnet

Globalnet is a Submariner add-on feature that allows you to connect clusters with overlapping Classless Inter-Domain Routings (CIDRs), without changing the CIDRs on existing clusters. Globalnet is a cluster set-wide configuration that you can select when you add the first managed cluster to a cluster set.

If you enable Globalnet, every managed cluster receives a global CIDR from the virtual Global Private Network, which is used to facilitate inter-cluster communication.

Important: You must enable Globalnet when clusters in a cluster set might have overlapping CIDRs.

The **ClusterAdmin** can enable Globalnet in the console by selecting the option **Enable Globalnet** when enabling the Submariner add-on for clusters in the cluster set. If you want to disable Globalnet after enabling it, you must first remove all managed clusters from your cluster set.

1.4.2.4.1. Enabling Globalnet by creating the *submariner-broker* object

When using the Red Hat Advanced Cluster Management APIs, the **ClusterAdmin** can enable Globalnet by creating a **submariner-broker** object in the **<ManagedClusterSet>-broker** namespace.

The **ClusterAdmin** role has the required permissions to create the **submariner-broker** object in the broker namespace. The **ManagedClusterSetAdmin** role, which is sometimes created to act as a proxy administrator for the cluster set, does not have the required permissions.

To provide the required permissions, the **ClusterAdmin** must associate the role permissions for the **access-to-brokers-submariner-crd** to the **ManagedClusterSetAdmin** user.

Complete the following steps to enable Globalnet by creating the **submariner-broker** object:

1. Retrieve the **<broker-namespace>** by running the following command:

```
oc get ManagedClusterSet <cluster-set-name> -o jsonpath="
{.metadata.annotations['cluster\.open-cluster-management\.io/submariner-broker-ns']}"
```

2. Create a **submariner-broker** object that specifies the Globalnet configuration by creating a YAML file named **submariner-broker**. Add content that resembles the following lines to the YAML file:

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
  name: submariner-broker 1
  namespace: broker-namespace 2
spec:
  globalnetEnabled: true-or-false 3
```

1. The name must be **submariner-broker**.
2. Replace **broker-namespace** with the name of your broker namespace.
3. Replace **true-or-false** with **true** to enable Globalnet.

3. Apply the file by running the following command:

```
oc apply -f submariner-broker.yaml
```

1.4.2.4.2. Configuring the number of global IPs

You can assign a configurable number of global IPs by changing the value of the **numberOfIPs** field in the **ClusterGlobalEgressIP** resource. The default value is 8. See the following example:

```
apiVersion: submariner.io/v1
kind: ClusterGlobalEgressIP
metadata:
  name: cluster-egress.submariner.io
spec:
  numberOfIPs: 8
```

1.4.2.4.3. Additional resources

- See the Submariner documentation to learn more about Submariner
- See Submariner NAT Traversal for more information about IP connectivity between gateway nodes.
- See the Submariner prerequisites documentation for more detailed information about the prerequisites.

- See Globalnet controller in the Submariner documentation for more information about Globalnet.

1.4.3. Installing the subctl command utility

The **subctl** utility is published on the Red Hat Developers page. To install the **subctl** utility locally, complete the following steps:

1. Go to the **subctl** publication directory.
2. Click the folder that matches the version of Submariner that you are using.
3. Click the **tar.xz** archive for the platform that you are using to download a compressed version of the **subctl** binary.
   a. If your platform is not listed, go to the Red Hat Ecosystem Catalog **subctl** page and extract **subctl** from the relevant image. For example, you can extract the *macos-arm64* binary from the *arm64* **subctl** image.
4. Decompress the **subctl** utility by entering the following command. Replace **<name>** with the name of the archive that you downloaded:

```
tar -C /tmp/ -xf <name>.tar.xz
```

5. Install the **subctl** utility by entering the following command. Replace **<name>** with the name of the archive that you downloaded. Replace **<version>** with the **subctl** version that you downloaded:

```
install -m744 /tmp/<version>/<name> /$HOME/.local/bin/subctl
```

Notes:

- Make sure that the **subctl** and Submariner versions match.
- For disconnected environments only, make sure to mirror the **submariner-nettest** image.

1.4.3.1. Using the subctl commands

After adding the utility to your path, view the following table for a brief description of the available commands:

export service	Creates a **ServiceExport** resource for the specified service, which enables other clusters in the Submariner deployment to discover the corresponding service.
unexport service	Removes the **ServiceExport** resource for the specified service, which prevents other clusters in the Submariner deployment from discovering the corresponding service.
show	Provides information about Submariner resources.

verify	Verifies connectivity, service discovery, and other Submariner features when Submariner is configured across a pair of clusters.
benchmark	Benchmarks throughput and latency across a pair of clusters that are enabled with Submariner or within a single cluster.
diagnose	Runs checks to identify issues that prevent the Submariner deployment from working correctly.
gather	Collects information from the clusters to help troubleshoot a Submariner deployment.
version	Displays the version details of the **subctl** binary tool.

Note: The Red Hat build of **subctl** only includes the commands that are relevant to Red Hat Advanced Cluster Management for Kubernetes. For more information about the **subctl** utility and its commands, see **subctl** in the Submariner documentation .

1.4.4. Deploying Submariner by using the console

Before you deploy Submariner with Red Hat Advanced Cluster Management for Kubernetes, you must prepare the clusters on the hosting environment. You can use the **SubmarinerConfig** API or the Red Hat Advanced Cluster Management for Kubernetes console to automatically prepare Red Hat OpenShift Container Platform clusters on the following providers:

- Amazon Web Services
- Google Cloud Platform
- IBM Power Systems Virtual Server
- Red Hat OpenShift on IBM Cloud (Technology Preview)
- Red Hat OpenStack Platform
- Microsoft Azure
- VMware vSphere

Notes:

- Only non-NSX deployments are supported on VMware vSphere.
- You must install the Calico API server on your cluster if you are using Red Hat OpenShift on IBM Cloud. Alternatively, you can manually create the the IP pools required for cross-cluster communication by following the CALICO CNI topic in the Submariner upstream documentation.

To deploy Submariner on other providers, follow the instructions in Deploying Submariner manually.

Complete the following steps to deploy Submariner with the Red Hat Advanced Cluster Management for Kubernetes console:

Required access: Cluster administrator

1. From the console, select **Infrastructure** > **Clusters**.
2. On the *Clusters* page, select the *Cluster sets* tab. The clusters that you want enable with Submariner must be in the same cluster set.
3. If the clusters on which you want to deploy Submariner are already in the same cluster set, skip to step 5.
4. If the clusters on which you want to deploy Submariner are not in the same cluster set, create a cluster set for them by completing the following steps:
   a. Select **Create cluster set**
   b. Name the cluster set, and select **Create**.
   c. Select **Manage resource assignments** to assign clusters to the cluster set.
   d. Select the managed clusters that you want to connect with Submariner to add them to the cluster set.
   e. Select **Review** to view and confirm the clusters that you selected.
   f. Select **Save** to save the cluster set, and view the resulting cluster set page.
5. On the cluster set page, select the *Submariner add-ons* tab.
6. Select **Install Submariner add-ons**.
7. Select the clusters on which you want to deploy Submariner.
8. See the fields in the following table and enter the required information in the *Install Submariner add-ons* editor:

Field	Notes
AWS Access Key ID	Only visible when you import an AWS cluster.
AWS Secret Access Key	Only visible when you import an AWS cluster.
Base domain resource group name	Only visible when you import an Azure cluster.
Client ID	Only visible when you import an Azure cluster.
Client secret	Only visible when you import an Azure cluster.
Subscription ID	Only visible when you import an Azure cluster.
Tenant ID	Only visible when you import an Azure cluster.

Field	Notes
Google Cloud Platform service account JSON key	Only visible when you import a Google Cloud Platform cluster.
Instance type	The instance type of the gateway node that is created on the managed cluster.
IPsec NAT-T port	The default value for the IPsec NAT traversal port is port **4500**. If your managed cluster environment is VMware vSphere, ensure that this port is opened on your firewall.
Gateway count	The number of gateway nodes to be deployed on the managed cluster. For AWS, GCP, Azure, and OpenStack clusters, dedicated Gateway nodes are deployed. For VWware clusters, existing worker nodes are tagged as gateway nodes. The default value is **1**. If the value is greater than 1, the Submariner gateway High Availability (HA) is automatically enabled.
Cable driver	The Submariner gateway cable engine component that maintains the cross-cluster tunnels. The default value is **Libreswan IPsec**.
Disconnected cluster	If enabled, tells Submariner to not access any external servers for public IP resolution.
Globalnet CIDR	Only visible when the Globalnet configuration is selected on the cluster set. The Globalnet CIDR to be used for the managed cluster. If left blank, a CIDR is allocated from the cluster set pool.

9. Select **Next** at the end of the editor to move to the editor for the next cluster, and complete the editor for each of the remaining clusters that you selected.

10. Verify your configuration for each managed cluster.

11. Click **Install** to deploy Submariner on the selected managed clusters.
It might take several minutes for the installation and configuration to complete. You can check the Submariner status in the list on the *Submariner add-ons* tab:

    - **Connection status** indicates how many Submariner connections are established on the managed cluster.
    - **Agent status** indicates whether Submariner is successfully deployed on the managed cluster. The console might report a status of **Degraded** until it is installed and configured.
    - **Gateway nodes labeled** indicates the number of gateway nodes on the managed cluster.

Submariner is now deployed on the selected clusters.

1.4.5. Deploying Submariner manually

Before you deploy Submariner with Red Hat Advanced Cluster Management for Kubernetes, you must prepare the clusters on the hosting environment for the connection. See Deploying Submariner by using the console to learn how to automatically deploy Submariner on supported clusters by using the console.

If your cluster is hosted on a provider that does not support automatic Submariner deployment, see the following sections to prepare the infrastructure manually. Each provider has unique steps for preparation, so make sure to select the correct provider.

1.4.5.1. Preparing bare metal for Submariner

To prepare bare metal clusters for deploying Submariner, complete the following steps:

1. Ensure that the firewall allows inbound/outbound traffic for external clients on the 4500/UDP and 4490/UDP ports for the Gateway nodes. Also, if the cluster is deployed with OpenShiftSDN CNI, allow inbound/outbound UDP/4800 traffic within the local cluster nodes.

2. Customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   gatewayConfig:
     gateways: 1
```

   Replace **managed-cluster-namespace** with the name of your managed cluster. The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

   This configuration labels one of the worker nodes as the Submariner gateway on your bare metal cluster.

   By default, Submariner uses IP security (IPsec) to establish the secure tunnels between the clusters on the gateway nodes. You can either use the default IPsec NATT port, or you can specify a different port that you configured. When you run this procedure without specifying an IPsec NATT port, 4500/UDP is used for the connections.

3. Identify the Gateway node configured by Submariner and enable firewall configurations to allow the IPsec NATT (UDP/4500) and NatDiscovery (UDP/4490) ports for external traffic.

See Customizing Submariner deployments for information about the customization options.

1.4.5.2. Preparing Microsoft Azure Red Hat OpenShift for Submariner by using the command line interface

The Microsoft Azure Red Hat OpenShift service combines various tools and resources that you can use to simplify the process of building container-based applications. To prepare Azure Red Hat OpenShift clusters for deploying Submariner by using the command line interface, complete the following steps:

1. Install the Azure CLI.

2. From the Azure CLI, run the following command to install the extension:

```
az extension add --upgrade -s <path-to-extension>
```

Replace **path-to-extension** with the path to where you downloaded the **.whl** extension file.

3. Run the following command to verify that the CLI extension is being used:

```
az extension list
```

If the extension is being used, the output might resemble the following example:

```
"experimental": false,
"extensionType": "whl",
"name": "aro",
"path": "<path-to-extension>",
"preview": true,
"version": "1.0.x"
```

4. From the Azure CLI, register the preview feature by running the following command:

```
az feature registration create --namespace Microsoft.RedHatOpenShift --name
AdminKubeconfig
```

5. Retrieve the administrator **kubeconfig** by running the following command:

```
az aro get-admin-kubeconfig -g <resource group> -n <cluster resource name>
```

Note: The **az aro** command saves the **kubeconfig** to the local directory and uses the name **kubeconfig**. To use it, set the environment variable **KUBECONFIG** to match the path of the file. See the following example:

```
export KUBECONFIG=<path-to-kubeconfig>
oc get nodes
```

6. Import your Azure Red Hat OpenShift cluster. See Cluster import introduction to learn more about how to import a cluster.

1.4.5.2.1. Preparing Microsoft Azure Red Hat OpenShift for Submariner by using the API

To prepare Azure Red Hat OpenShift clusters for deploying Submariner by using the API, customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  loadBalancerEnable: true
```

Replace **managed-cluster-namespace** with the name of your managed cluster.

The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

This configuration labels one of the worker nodes as the Submariner gateway on your Azure Red Hat OpenShift cluster.

By default, Submariner uses IP security (IPsec) to establish the secure tunnels between the clusters on the gateway nodes. You can either use the default IPsec NATT port, or you can specify a different port that you configured. When you run this procedure without specifying an IPsec NATT port, port 4500/UDP is used for the connections.

See Customizing Submariner deployments for information about the customization options.

1.4.5.3. Preparing Red Hat OpenShift Service on AWS for Submariner by using the command line interface

Red Hat OpenShift Service on AWS provides a stable and flexible platform for application development and modernization. To prepare OpenShift Service on AWS clusters for deploying Submariner, complete the following steps:

1. Log in to OpenShift Service on AWS by running the following commands:

```
rosa login
oc login <rosa-cluster-url>:6443 --username cluster-admin --password <password>
```

2. Create a **kubeconfig** for your OpenShift Service on AWS cluster by running the following command:

```
oc config view --flatten=true > rosa_kube/kubeconfig
```

3. Import your OpenShift Service on AWS cluster. See Cluster import introduction to learn more about how to import a cluster.

1.4.5.3.1. Preparing Red Hat OpenShift Service on AWS for Submariner by using the API

To prepare OpenShift Service on AWS clusters for deploying Submariner by using the API, customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   loadBalancerEnable: true
```

Replace **managed-cluster-namespace** with the name of your managed cluster.

The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

By default, Submariner uses IP security (IPsec) to establish the secure tunnels between the clusters on the gateway nodes. You can either use the default IPsec NATT port, or you can specify a different port that you configured. When you run this procedure without specifying an IPsec NATT port, port 4500/UDP is used for the connections.

See Customizing Submariner deployments for information about the customization options.

1.4.5.4. Deploy Submariner with the ManagedClusterAddOn API

After manually preparing your selected hosting environment, you can deploy Submariner with the **ManagedClusterAddOn** API by completing the following steps:

1. Create a **ManagedClusterSet** resource on the hub cluster by using the instructions provided in the Creating a *ManagedClusterSet* documentation. Make sure your entry for the **ManagedClusterSet** resembles the following content:

```
apiVersion: cluster.open-cluster-management.io/v1beta2
kind: ManagedClusterSet
metadata:
  name: <managed-cluster-set-name>
```

   Replace **managed-cluster-set-name** with a name for the **ManagedClusterSet** that you are creating.

   Important: The maximum character length of a Kubernetes namespace is 63 characters. The maximum character length you can use for the **<managed-cluster-set-name>** is 56 characters. If the character length of **<managed-cluster-set-name>** exceeds 56 characters, the **<managed-cluster-set-name>** is cut off from the head.

   After the **ManagedClusterSet** is created, the **submariner-addon** creates a namespace called **<managed-cluster-set-name>-broker** and deploys the Submariner broker to it.

2. Create the **Broker** configuration on the hub cluster in the **<managed-cluster-set-name>-broker** namespace by customizing and applying YAML content that is similar to the following example:

```
apiVersion: submariner.io/v1alpha1
kind: Broker
metadata:
    name: submariner-broker
    namespace: <managed-cluster-set-name>-broker
    labels:
       cluster.open-cluster-management.io/backup: submariner
spec:
    globalnetEnabled: <true-or-false>
```

   Replace **managed-cluster-set-name** with the name of the managed cluster.

   Set the value of **globalnetEnabled** to **true** if you want to enable Submariner Globalnet in the **ManagedClusterSet**.

3. Add one managed cluster to the **ManagedClusterSet** by running the following command:

```
oc label managedclusters <managed-cluster-name> "cluster.open-cluster-
management.io/clusterset=<managed-cluster-set-name>" --overwrite
```

   Replace **<managed-cluster-name>** with the name of the managed cluster that you want to add to the **ManagedClusterSet**.

   Replace **<managed-cluster-set-name>** with the name of the **ManagedClusterSet** to which you want to add the managed cluster.

4. Customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:{}
```

Replace **managed-cluster-namespace** with the namespace of your managed cluster.

Note: The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

5. Deploy Submariner on the managed cluster by customizing and applying YAML content that is similar to the following example:

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

Replace **managed-cluster-name** with the name of the managed cluster that you want to use with Submariner.

The **installNamespace** field in the spec of the **ManagedClusterAddOn** is the namespace on the managed cluster where it installs Submariner. Currently, Submariner must be installed in the **submariner-operator** namespace.

After the **ManagedClusterAddOn** is created, the **submariner-addon** deploys Submariner to the **submariner-operator** namespace on the managed cluster. You can view the deployment status of Submariner from the status of this **ManagedClusterAddOn**.

Note: The name of **ManagedClusterAddOn** must be **submariner**.

6. Repeat steps three, four, and five for all of the managed clusters that you want to enable Submariner on.

7. After Submariner is deployed on the managed cluster, you can verify the Submariner deployment status by checking the status of submariner **ManagedClusterAddOn** by running the following command:

```
oc -n <managed-cluster-name> get managedclusteraddons submariner -oyaml
```

Replace **managed-cluster-name** with the name of the managed cluster.

In the status of the Submariner **ManagedClusterAddOn**, three conditions indicate the deployment status of Submariner:

- **SubmarinerGatewayNodesLabeled** condition indicates whether there are labeled Submariner gateway nodes on the managed cluster.

- **SubmarinerAgentDegraded** condition indicates whether the Submariner is successfully deployed on the managed cluster.

- **SubmarinerConnectionDegraded** condition indicates how many connections are established on the managed cluster with Submariner.

1.4.6. Customizing Submariner deployments

You can customize some of the settings of your Submariner deployments, including your Network Address Translation-Traversal (NATT) port, number of gateway nodes, and instance type of your gateway nodes. These customizations are consistent across all of the providers.

1.4.6.1. NATT port

If you want to customize your NATT port, customize and apply the following YAML content for your provider environment:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
   name: submariner
   namespace: <managed-cluster-namespace>
spec:
   credentialsSecret:
     name: <managed-cluster-name>-<provider>-creds
   IPSecNATTPort: <NATTPort>
```

- Replace **managed-cluster-namespace** with the namespace of your managed cluster.
- Replace **managed-cluster-name** with the name of your managed cluster
  - AWS: Replace **provider** with **aws**. The value of **<managed-cluster-name>-aws-creds** is your AWS credential secret name, which you can find in the cluster namespace of your hub cluster.
  - GCP: Replace **provider** with **gcp**. The value of **<managed-cluster-name>-gcp-creds** is your Google Cloud Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
  - OpenStack: Replace **provider** with **osp**. The value of **<managed-cluster-name>-osp-creds** is your Red Hat OpenStack Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
  - Azure: Replace **provider** with **azure**. The value of **<managed-cluster-name>-azure-creds** is your Microsoft Azure credential secret name, which you can find in the cluster namespace of your hub cluster.
- Replace **managed-cluster-namespace** with the namespace of your managed cluster.
- Replace **managed-cluster-name** with the name of your managed cluster. The value of **managed-cluster-name-gcp-creds** is your Google Cloud Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
- Replace **NATTPort** with the NATT port that you want to use.

Note: The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

1.4.6.2. Number of gateway nodes

If you want to customize the number of your gateway nodes, customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
 gatewayConfig:
    gateways: <gateways>
```

- Replace **managed-cluster-namespace** with the namespace of your managed cluster.
- Replace **managed-cluster-name** with the name of your managed cluster.
  - AWS: Replace **provider** with **aws**. The value of **<managed-cluster-name>-aws-creds** is your AWS credential secret name, which you can find in the cluster namespace of your hub cluster.
  - GCP: Replace **provider** with **gcp**. The value of **<managed-cluster-name>-gcp-creds** is your Google Cloud Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
  - OpenStack: Replace **provider** with **osp**. The value of **<managed-cluster-name>-osp-creds** is your Red Hat OpenStack Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
  - Azure: Replace **provider** with **azure**. The value of **<managed-cluster-name>-azure-creds** is your Microsoft Azure credential secret name, which you can find in the cluster namespace of your hub cluster.
- Replace **gateways** with the number of gateways that you want to use. If the value is greater than 1, the Submariner gateway automatically enables high availability.

Note: The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

1.4.6.3. Instance types of gateway nodes

If you want to customize the instance type of your gateway node, customize and apply YAML content that is similar to the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
 gatewayConfig:
    instanceType: <instance-type>
```

- Replace **managed-cluster-namespace** with the namespace of your managed cluster.

- Replace **managed-cluster-name** with the name of your managed cluster.
    - AWS: Replace **provider** with **aws**. The value of **<managed-cluster-name>-aws-creds** is your AWS credential secret name, which you can find in the cluster namespace of your hub cluster.
    - GCP: Replace **provider** with **gcp**. The value of **<managed-cluster-name>-gcp-creds** is your Google Cloud Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
    - OpenStack: Replace **provider** with **osp**. The value of **<managed-cluster-name>-osp-creds** is your Red Hat OpenStack Platform credential secret name, which you can find in the cluster namespace of your hub cluster.
    - Azure: Replace **provider** with **azure**. The value of **<managed-cluster-name>-azure-creds** is your Microsoft Azure credential secret name, which you can find in the cluster namespace of your hub cluster.
- Replace **instance-type** with the AWS instance type that you want to use.

Note: The name of the **SubmarinerConfig** must be **submariner**, as shown in the example.

1.4.6.4. Cable driver

The Submariner Gateway Engine component creates secure tunnels to other clusters. The cable driver component maintains the tunnels by using a pluggable architecture in the Gateway Engine component. You can use the Libreswan or VXLAN implementations for the **cableDriver** configuration of the cable engine component. See the following example:

```
apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
kind: SubmarinerConfig
metadata:
  name: submariner
  namespace: <managed-cluster-namespace>
spec:
  cableDriver: vxlan
  credentialsSecret:
    name: <managed-cluster-name>-<provider>-creds
```

Best practice: Do not use the VXLAN cable driver on public networks. The VXLAN cable driver is unencrypted. Only use VXLAN to avoid unnecessary double encryption on private networks. For example, some on-premise environments might handle the tunnel's encryption with a dedicated line-level hardware device.

1.4.6.5. Using a customized Submariner subscription

The Submariner add-on automatically configures a subscription for Submariner; this ensures that the version of Submariner appropriate for the installed version of Red Hat Advanced Cluster Management is installed and kept up-to-date. If you want to change this behavior, or if you want to manually control Submariner upgrades, you can customize the Submariner subscription.

When you use a customized Submariner subscription, you must complete the following fields:

- **Source:** The catalog source to use for the Submariner subscription. For example, **redhat-operators**.

- **source Namespace:** The namespace of the catalog source. For example, **openshift-marketplace**.
- **Channel:** The channel to follow for the subscription. For example, for Red Hat Advanced Cluster Management 2.12, **stable-0.19**.
- **Starting CSV (Optional):** The initial **ClusterServiceVersion**.
- **Install Plan Approval:** The decision to manually or automatically approve install plans.

Note: If you want to manually approve the install plan, you must use a customized Submariner subscription.

1.4.7. Managing service discovery for Submariner

After Submariner is deployed into the same environment as your managed clusters, the routes are configured for secure IP routing between the pod and services across the clusters in the managed cluster set.

1.4.7.1. Enabling service discovery for Submariner

To make a service from a cluster visible and discoverable to other clusters in the managed cluster set, you must create a **ServiceExport** object. After a service is exported with a **ServiceExport** object, you can access the service by the following format: **<service>.<namespace>.svc.clusterset.local**. If multiple clusters export a service with the same name, and from the same namespace, they are recognized by other clusters as a single logical service.

This example uses the **nginx** service in the **default** namespace, but you can discover any Kubernetes **ClusterIP** service or headless service:

1. Apply an instance of the **nginx** service on a managed cluster that is in the **ManagedClusterSet** by entering the following commands:

```
oc -n default create deployment nginx --image=nginxinc/nginx-unprivileged:stable-alpine
oc -n default expose deployment nginx --port=8080
```

2. Export the service by creating a **ServiceExport** entry by entering a command with the **subctl** tool that is similar to the following command:

```
subctl export service --namespace <service-namespace> <service-name>
```

   Replace **service-namespace** with the name of the namespace where the service is located. In this example, it is **default**.

   Replace **service-name** with the name of the service that you are exporting. In this example, it is **nginx**.

   See **export** in the Submariner documentation for more information about other available flags.

3. Run the following command from a different managed cluster to confirm that it can access the **nginx** service:

```
oc -n default run --generator=run-pod/v1 tmp-shell --rm -i --tty --image
quay.io/submariner/nettest -- /bin/bash curl nginx.default.svc.clusterset.local:8080
```

The **nginx** service discovery is now configured for Submariner.

1.4.7.2. Disabling service discovery for Submariner

To disable a service from being exported to other clusters, enter a command similar to the following example for **nginx**:

```
subctl unexport service --namespace <service-namespace> <service-name>
```

Replace **service-namespace** with the name of the namespace where the service is located.

Replace **service-name** with the name of the service that you are exporting.

See **unexport** in the Submariner documentation for more information about other available flags.

The service is no longer available for discovery by clusters.

1.4.8. Uninstalling Submariner

You can uninstall the Submariner components from your clusters using the Red Hat Advanced Cluster Management for Kubernetes console or the command-line. For Submariner versions earlier than 0.12, additional steps are needed to completely remove all data plane components. The Submariner uninstall is idempotent, so you can repeat steps without any issues.

1.4.8.1. Uninstalling Submariner by using the console

To uninstall Submariner from a cluster by using the console, complete the following steps:

1. From the console navigation, select **Infrastructure** > **Clusters**, and select the *Cluster sets* tab.
2. Select the cluster set that contains the clusters from which you want to remove the Submariner components.
3. Select the **Submariner Add-ons** tab to view the clusters in the cluster set that have Submariner deployed.
4. In the *Actions* menu for the cluster that you want to uninstall Submariner, select **Uninstall Add-on**.
5. In the *Actions* menu for the cluster that you want to uninstall Submariner, select **Delete cluster sets**.
6. Repeat those steps for other clusters from which you are removing Submariner.
   Tip: You can remove the Submariner add-on from multiple clusters in the same cluster set by selecting multiple clusters and clicking **Actions**. Select **Uninstall Submariner add-ons**.

If the version of Submariner that you are removing is earlier than version 0.12, continue with Uninstalling Submariner manually. If the Submariner version is 0.12 or later, Submariner is removed.

Important: Verify that all of the cloud resources are removed from the cloud provider to avoid additional charges by your cloud provider. See Verifying Submariner resource removal for more information.

1.4.8.2. Uninstalling Submariner by using the CLI

To uninstall Submariner by using the command line, complete the following steps:

1. Remove the Submariner deployment for the cluster by running the following command:

```
oc -n <managed-cluster-namespace> delete managedclusteraddon submariner
```

   Replace **managed-cluster-namespace** with the namespace of your managed cluster.

2. Remove the cloud resources of the cluster by running the following command:

```
oc -n <managed-cluster-namespace> delete submarinerconfig submariner
```

   Replace **managed-cluster-namespace** with the namespace of your managed cluster.

3. Delete the cluster set to remove the broker details by running the following command:

```
oc delete managedclusterset <managedclusterset>
```

   Replace **managedclusterset** with the name of your managed cluster set.

If the version of Submariner that you are removing is earlier than version 0.12, continue with Uninstalling Submariner manually. If the Submariner version is 0.12 or later, Submariner is removed.

Important: Verify that all of the cloud resources are removed from the cloud provider to avoid additional charges by your cloud provider. See Verifying Submariner resource removal for more information.

1.4.8.3. Uninstalling Submariner manually

When uninstalling versions of Submariner that are earlier than version 0.12, complete steps 5-8 in the Manual Uninstall section in the Submariner documentation.

After completing those steps, your Submariner components are removed from the cluster.

Important: Verify that all of the cloud resources are removed from the cloud provider to avoid additional charges by your cloud provider. See Verifying Submariner resource removal for more information.

1.4.8.4. Verifying Submariner resource removal

After uninstalling Submariner, verify that all of the Submariner resources are removed from your clusters. If they remain on your clusters, some resources continue to accrue charges from infrastructure providers. Ensure that you have no additional Submariner resourceson your cluster by completing the following steps:

1. Run the following command to list any Submariner resources that remain on the cluster:

```
oc get cluster <CLUSTER_NAME> grep submariner
```

   Replace **CLUSTER_NAME** with the name of your cluster.

2. Remove any resources on the list by entering the following command:

```
oc delete resource <RESOURCE_NAME> cluster <CLUSTER_NAME>
```

   Replace **RESOURCE_NAME** with the name of the Submariner resource that you want to remove.

3. Repeat steps 1-2 for each of the clusters until your search does not identify any resources.

The Submariner resources are removed from your cluster.