



**Red Hat**

# Red Hat Advanced Cluster Management for Kubernetes 2.8

## Networking

Networking





## Legal Notice

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## Abstract

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

# Table of Contents

<b>CHAPTER 1. NETWORKING</b> .....	<b>3</b>
1.1. HUB CLUSTER NETWORK CONFIGURATION	3
1.1.1. Hub cluster network configuration table	3
1.2. MANAGED CLUSTER NETWORK CONFIGURATION	5
1.2.1. Managed cluster network configuration table	5
1.3. ADVANCED NETWORK CONFIGURATION	7
1.3.1. Additional networking requirements for infrastructure operator table	7
1.3.2. Submariner networking requirements table	7
1.3.3. Additional networking requirements for Hive table	8
1.3.4. Hosted control planes networking requirements table (Technology Preview)	8
1.3.5. Application deployment network requirements table	9
1.3.6. Namespace connection network requirements table	9
1.4. SUBMARINER MULTICLUSTER NETWORKING AND SERVICE DISCOVERY	10
1.4.1. Deploying Submariner on disconnected clusters	10
1.4.1.1. Configuring Submariner on disconnected clusters	10
1.4.1.1.1. Mirroring images in the local registry	11
1.4.1.1.2. Customizing catalogSource names	11
1.4.1.1.3. Enabling airGappedDeployment in SubmarinerConfig	11
1.4.2. Configuring Submariner	11
1.4.2.1. Prerequisites	11
1.4.2.2. Submariner ports table	12
1.4.2.3. Globalnet	12
1.4.3. Installing the subctl command utility	13
1.4.3.1. Using the subctl commands	13
1.4.4. Deploying Submariner by using the console	14
1.4.5. Deploying Submariner manually	17
1.4.5.1. Preparing bare metal for Submariner	17
1.4.5.2. Preparing Microsoft Azure Red Hat OpenShift for Submariner by using the console	17
1.4.5.2.1. Preparing Microsoft Azure Red Hat OpenShift for Submariner by using the API	18
1.4.5.3. Preparing Red Hat OpenShift Service on AWS for Submariner by using the console	19
1.4.5.3.1. Preparing Red Hat OpenShift Service on AWS for Submariner by using the API	20
1.4.5.4. Deploy Submariner with the ManagedClusterAddOn API	20
1.4.6. Customizing Submariner deployments	22
1.4.6.1. NATT port	22
1.4.6.2. Number of gateway nodes	23
1.4.6.3. Instance types of gateway nodes	24
1.4.6.4. Cable driver	25
1.4.7. Managing service discovery for Submariner	25
1.4.7.1. Enabling service discovery for Submariner	25
1.4.7.2. Disabling service discovery for Submariner	26
1.4.8. Uninstalling Submariner	26
1.4.8.1. Uninstalling Submariner by using the console	26
1.4.8.2. Uninstalling Submariner by using the CLI	27
1.4.8.3. Uninstalling Submariner manually	27
1.4.8.4. Verifying Submariner resource removal	27



# 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 <b>klusterlet-addon-workmgr</b> 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 ArgoCD 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 OpenShift Container Platform version 4.8, or later	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 <b>klusterlet-addon-workmgr</b> 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 OpenShift Container Platform version 4.8, or later	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 ArgoCD 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](#)
- [Hosted control planes networking requirements table \(Technology Preview\)](#)
- [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 <b>libvirt</b> provisioning host	IP	Connects the hub cluster, where the Hive operator is installed, to the <b>libvirt</b> 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. Hosted control planes networking requirements table (Technology Preview)

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

Direction	Connection	Port (if specified)
Outbound	OpenShift Container Platform control-plane and worker nodes	
Outbound	For hosted clusters on Amazon Web Services only: Outbound connection to AWS API and S3 API	
Outbound	For hosted clusters on Microsoft Azure cloud services only: Outbound connection to Azure API	

Direction	Connection	Port (if specified)
Outbound	OpenShift Container Platform image repositories that store the ISO images of the coreOS and the image registry for OpenShift Container Platform pods	
Outbound	Local API client of the klusterlet on the hosting cluster communicates with the API of the HyperShift hosted cluster	

### 1.3.5. 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

### 1.3.6. 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 can be used 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](#).

Make sure to see 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. You can find more information about Submariner in the [Submariner open source project documentation](#).

#### 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 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 at Red Hat OpenShift Container Platform version 4.11 or later.
- If your Red Hat 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.
- 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 which Submariner ports you need to 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

See the [Submariner upstream prerequisites documentation](#) for more detailed information about the prerequisites.

### 1.4.2.3. Globalnet

Globalnet is a feature included with the Submariner add-on which supports connectivity between clusters with overlapping CIDRs. Globalnet is a cluster set wide configuration, and can be selected when the first managed cluster is added to the cluster set. When Globalnet is enabled, each managed cluster is allocated a global CIDR from the virtual Global Private Network. The global CIDR is used for supporting inter-cluster communication.

If there is a chance that your clusters running Submariner might have overlapping CIDRs, consider enabling Globalnet. When using the console, the **ClusterAdmin** can enable Globalnet for a cluster set by selecting the option **Enable Globalnet** when enabling the Submariner add-on for clusters in the cluster set. After you enable Globalnet, you cannot disable it without removing Submariner.

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 this 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 create 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
  namespace: <broker-namespace>
spec:
  globalnetEnabled: <true-or-false>
```

Replace **broker-namespace** with the name of your broker namespace.

Replace **true-or-false** with **true** to enable Globalnet.

**Note:** The **metadata name** parameter must be **submariner-broker**.

3. Apply the file to your YAML file by entering the following command:

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

For more information about Globalnet, see [Globalnet controller](#) in the Submariner documentation.

### 1.4.3. Installing the subctl command utility

The **subctl** utility is shipped in a container image. Complete the following steps to install the **subctl** utility locally:

1. Log in to the registry by running the following command and entering your credentials when prompted:

```
oc registry login --registry registry.redhat.io
```

2. Download the [subctl container](#) and extract a compressed version of the **subctl** binary to **/tmp** by entering the following command:

```
oc image extract registry.redhat.io/rhacm2/subctl-rhel8:v0.15 --path="/dist/subctl-v0.15*-linux-amd64.tar.xz":/tmp/ --confirm
```

3. Decompress the **subctl** utility by entering the following command:

```
tar -C /tmp/ -xf /tmp/subctl-v0.15*-linux-amd64.tar.xz
```

4. Install the **subctl** utility by entering the following command:

```
install -m744 /tmp/subctl-v0.15*/subctl-v0.15*-linux-amd64 /$HOME/.local/bin/subctl
```

#### 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:

<b>export service</b>	Creates a <b>ServiceExport</b> resource for the specified service, which enables other clusters in the Submariner deployment to discover the corresponding service.
<b>unexport service</b>	Removes the <b>ServiceExport</b> resource for the specified service, which prevents other clusters in the Submariner deployment from discovering the corresponding service.
<b>show</b>	Provides information about Submariner resources.
<b>verify</b>	Verifies connectivity, service discovery, and other Submariner features when Submariner is configured across a pair of clusters.
<b>benchmark</b>	Benchmarks throughput and latency across a pair of clusters that are enabled with Submariner or within a single cluster.
<b>diagnose</b>	Runs checks to identify issues that prevent the Submariner deployment from working correctly.
<b>gather</b>	Collects information from the clusters to help troubleshoot a Submariner deployment.
<b>version</b>	Displays the version details of the <b>subctl</b> binary tool.

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
- Red Hat OpenStack Platform
- Microsoft Azure
- VMware vSphere

**Note:** Only non-NSX deployments are supported on VMware vSphere.

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
<b>AWS Access Key ID</b>	Only visible when you import an AWS cluster.
<b>AWS Secret Access Key</b>	Only visible when you import an AWS cluster.
<b>Base domain resource group name</b>	Only visible when you import an Azure cluster.
<b>Client ID</b>	Only visible when you import an Azure cluster.
<b>Client secret</b>	Only visible when you import an Azure cluster.
<b>Subscription ID</b>	Only visible when you import an Azure cluster.
<b>Tenant ID</b>	Only visible when you import an Azure cluster.

Field	Notes
<b>Google Cloud Platform service account JSON key</b>	Only visible when you import a Google Cloud Platform cluster.
<b>Instance type</b>	The instance type of the gateway node that is created on the managed cluster.
<b>IPsec NAT-T port</b>	The default value for the IPsec NAT traversal port is port <b>4500</b> . If your managed cluster environment is VMware vSphere, ensure that this port is opened on your firewall.
<b>Gateway count</b>	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 VMware clusters, existing worker nodes are tagged as gateway nodes. The default value is <b>1</b> . If the value is greater than 1, the Submariner gateway High Availability (HA) is automatically enabled.
<b>Cable driver</b>	The Submariner gateway cable engine component that maintains the cross-cluster tunnels. The default value is <b>Libreswan IPsec</b> .
<b>Disconnected cluster</b>	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 be 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 console

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

1. Download the [Python wheel and CLI extension](#).
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.

- 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"
```

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

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

- 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
```

- Import your Azure Red Hat OpenShift cluster to your cluster list by selecting **Infrastructure > Clusters > Import cluster** from the Red Hat Advanced Cluster Management console.
- Select the **Kubeconfig Import mode** and enter the content from your **kubeconfig** file in the **Kubeconfig** window. Follow the instructions in the console to complete the import. You can verify that your Azure Red Hat OpenShift cluster was imported successfully by navigating to **Infrastructure > Clusters**.
- Navigate to **Infrastructure > Clusters > Cluster sets** and select the name of the cluster set that you want to add. Then, click the **Submariner add-ons** tab.
- Click the **Install Submariner add-ons** button and set your Azure Red Hat OpenShift cluster as your **Target clusters**. Follow the instructions in the console to complete the install.
- Navigate to **Infrastructure > Clusters > Cluster sets > Submariner add-ons** to verify that your Azure Red Hat OpenShift cluster **Connection status** is **Healthy**.

#### 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 console

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. Create a new node to run Submariner gateway by running the following command:

```

rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number
of Submariner gateway > --labels='submariner.io/gateway=true'

```

2. 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>

```

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

```

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

```

4. Import your OpenShift Service on AWS cluster to your cluster list by selecting **Infrastructure > Clusters > Import cluster** from the Red Hat Advanced Cluster Management console.
5. Select the **Kubeconfig Import mode** and enter the content from your **kubeconfig** file in the **Kubeconfig** window. Follow the instructions in the console to complete the import. You can verify that your OpenShift Service on AWS cluster was imported successfully by navigating to **Infrastructure > Clusters**.
6. Navigate to **Infrastructure > Clusters > Cluster sets** and select the name of the cluster set that you want to add. Then, click the **Submariner add-ons** tab.

7. Click the **Install Submariner add-ons** button and set your OpenShift Service on AWS cluster as your **Target clusters**. Follow the instructions in the console to complete the installation.
8. Navigate to **Infrastructure > Clusters > Cluster sets > Submariner add-ons** to verify that your OpenShift Service on AWS cluster **Connection status** is **Healthy**.

#### 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, complete the following steps:

1. Create a new node to run Submariner gateway by running the following command:

```
rosa create machinepool --cluster=<cluster_name> --name=sm-gw-mp --replicas=<number of Submariner gateway > --labels='submariner.io/gateway=true'
```

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:
  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: submarineraddn.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.

#### 1.4.7. 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.cluster.set.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**.

- 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:

- 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.

- 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.

- 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 resources on 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.