Red Hat Advanced Cluster Management for Kubernetes 2.4

Services

Read more to learn how to add supported services to your cluster.

Last Updated: 2022-12-09

Red Hat Advanced Cluster Management for Kubernetes 2.4 Services

Read more to learn how to add supported services to your cluster.

Legal Notice

Abstract

Read more to learn how to add supported services to your cluster.

Table of Contents

CHAPTER 1. SERVICES OVERVIEW

You can add services to use with Red Hat Advanced Cluster Management for Kubernetes that can improve some area of performance. A service runs on a single managed cluster or on multiple managed clusters.

The following sections provide a summary of the service that is available for Red Hat Advanced Cluster Management:

- Deploying Submariner with the console
- Deploying Submariner with APIs
- Enabling service discovery for Submariner

1.1. SUBMARINER NETWORKING SERVICE (TECHNOLOGY PREVIEW)

The **submariner-addon** component is a **Technology Preview** feature.

The *Submariner* service is an open source tool that can be used with Red Hat Advanced Cluster Management for Kubernetes to provide direct networking between pods across two or more Kubernetes clusters in your environment, either on-premises or in the cloud. For more information about Submariner, see Submariner's website.

You can enable Submariner on OpenShift Container Platform clusters that are hosted in the following environments:

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- IBM Cloud
- Microsoft Azure
- Red Hat OpenShift Dedicated
- VMware vSphere

Red Hat Advanced Cluster Management for Kubernetes provides a Submariner component that you can deploy in your environment by using your hub cluster.

- Prerequisites
- Preparing selected hosts to deploy Submariner
  - Preparing Microsoft Azure to deploy Submariner
  - Preparing IBM Cloud to deploy Submariner
  - Preparing Red Hat OpenShift Dedicated to deploy Submariner

1.1.1. Prerequisites

Ensure that you have the following prerequisites before using Submariner:

- A Red Hat Advanced Cluster Management hub cluster that is running on Red Hat OpenShift Container Platform version 4.6, or later, with Kubernetes version 1.19, or later.

- A credential to access the hub cluster with **cluster-admin** permissions.

- Two or more OpenShift Container Platform managed clusters that are running on OpenShift Container Platform version 4.4, or later, with Kubernetes version 1.17, or later, and are managed by the Red Hat Advanced Cluster Management hub cluster.

- Non-overlapping pod and service Classless Inter-Domain Routing (CIDR) ranges for clusters. **Note:** The upstream feature called *Globalnet* allows overlapping CIDRs, but it is not currently supported for use with Red Hat Advanced Cluster Management.

- 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.

- Firewall configuration across all nodes in each of the managed clusters must allow 4800/UDP in both directions.

- Firewall configuration on the Gateway nodes must allow ingress 8080/TCP so the other nodes in the cluster can access it.

- Firewall configuration open for 4500/UDP and any other ports that are used for IPsec traffic on the gateway nodes.
  Note: This is configured automatically when your clusters are deployed in an AWS or GCP environment, but must be configured manually for clusters on other environments and for the firewalls that protect private clouds.

  Table 1.1. Submariner required ports

Name	Default value	Customizable
IPsec NATT	4500/UDP	Yes
VXLAN	4800/UDP	No
Submariner metrics port	8080/TCP	No

See Submariner's upstream prerequisites docs for more detailed information about the prerequisites.

1.1.2. Preparing selected hosts to deploy Submariner

Before you deploy Submariner with Red Hat Advanced Cluster Management for Kubernetes on Microsoft Azure, IBM Cloud, or Red Hat OpenShift Dedicated, you must manually prepare the clusters on the hosting environment for the connection. The requirements are different for different hosting environments, so follow the instructions for your hosting environment.

1.1.2.1. Preparing Microsoft Azure to deploy Submariner

To prepare the clusters on your Microsoft Azure for deploying the Submariner component, complete the following steps:

1. Create the inbound and outbound firewall rules on your Microsoft Azure environment to

1. Create the inbound and outbound firewall rules on your Microsoft Azure environment to open the IP security NAT traversal ports (by default 4500/UDP) to enable Submariner communication:

    ```
    # Create inbound NAT rule
    $ az network lb inbound-nat-rule create --lb-name <lb-name> \
    --resource-group <res-group> \
    --name <name> \
    --protocol Udp --frontend-port <ipsec-port> \
    --backend-port <ipsec-port> \
    --frontend-ip-name <frontend-ip-name>

    # Add VM network interface to the just-created inbound NAT rule
    $ az network nic ip-config inbound-nat-rule add \
    --lb-name <lb-name> --resource-group <res-group> \
    --inbound-nat-rule <nat-name> \
    --nic-name <nic-name> --ip-config-name <pipConfig>
    ```

    Replace **lb-name** with your load balancer name.

    Replace **res-group** with your resource group name.

    Replace **nat-name** with your load balancing inbound NAT rule name.

    Replace **ipsec-port** with your IPsec port.

    Replace **pipConfig** with your cluster frontend IP configuration name.

    Replace **nic-name** with your network interface card (NIC) name.

2. Create one load balancing inbound NAT rules to forward Submariner gateway metrics service request:

    ```
    # Create inbound NAT rule
    $ az network lb inbound-nat-rule create --lb-name <lb-name> \
    --resource-group <res-group> \
    --name <name> \
    --protocol Tcp --frontend-port 8080 --backend-port 8080 \
    --frontend-ip-name <frontend-ip-name>

    # Add VM network interface to the just-created inbound NAT rule
    $ az network nic ip-config inbound-nat-rule add \
    --lb-name <lb-name> --resource-group <res-group> \
    --inbound-nat-rule <nat-name> \
    --nic-name <nic-name> --ip-config-name <pipConfig>
    ```

    Replace **lb-name** with your load balancer name.

    Replace **res-group** with your resource group name.

    Replace **nat-name** with your load balancing inbound NAT rule name.

    Replace **pipConfig** with your cluster frontend IP configuration name.

    Replace **nic-name** with your network interface card (NIC) name.

3. Create network security groups (NSG) security rules on your Azure to open a NAT traversal port (by default 4500/UDP) for Submariner:

   ```
   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Inbound --access Allow \
   --protocol Udp --destination-port-ranges <ipsec-port>

   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Outbound --access Allow \
   --protocol Udp --destination-port-ranges <ipsec-port>
   ```

   Replace **res-group** with your resource group name.

   Replace **nsg-name** with your NSG name.

   Replace **priority** with your rule priority.

   Replace **name** with your rule name.

   Replace **ipsec-port** with your IPsec port.

4. Create the NSG rules to open 4800/UDP port to encapsulate pod traffic from the worker and master nodes to the Submariner Gateway nodes:

   ```
   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Inbound --access Allow \
   --protocol Udp --destination-port-ranges 4800 \

   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Outbound --access Allow \
   --protocol Udp --destination-port-ranges 4800
   ```

   Replace **res-group** with your resource group name.

   Replace **nsg-name** with your NSG name.

   Replace **priority** with your rule priority.

   Replace **name** with your rule name.

5. Create the NSG rules to open 8080/TCP port to export metrics service from the Submariner Gateway nodes:

   ```
   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Inbound --access Allow \
   --protocol Tcp --destination-port-ranges 8080 \

   $ az network nsg rule create --resource-group <res-group> \
   --nsg-name <nsg-name> --priority <priority> \
   --name <name> --direction Outbound --access Allow \
   --protocol Udp --destination-port-ranges 8080
   ```

Replace **res-group** with your resource group name.

Replace **nsg-name** with your NSG name.

Replace **priority** with your rule priority.

Replace **name** with your rule name.

6. Label a worker node in the cluster with the following label: **submariner.io/gateway=true**.

1.1.2.2. Preparing IBM Cloud to deploy Submariner

There are two kinds of Red Hat OpenShift Kubernetes Service (ROKS) on IBM Cloud: the classic cluster and the second generation of compute infrastructure in a virtual private cloud (VPC). Submariner cannot run on the classic ROKS cluster since it cannot configure the IPsec ports for the classic cluster.

To configure the ROKS clusters on a VPC to use Submariner, complete the steps in the following links:

1. Before you create a cluster, specify subnets for pods and services, which avoids overlapping CIDRs with other clusters. Make sure there are no overlapping pods and services CIDRs between clusters if you are using an existing cluster. See VPC Subnets for the procedure.

2. Attach a public gateway to subnets used in the cluster. See Public Gateway for the procedure.

3. Create inbound rules for the default security group of the cluster by completing the steps in Security Group . Ensure that the firewall allows inbound and outbound traffic on 4500/UDP and 500/UDP ports for Gateway nodes, and allows inbound and outbound UDP/4800 for all the other nodes.

4. Label a node that has the public gateway as **submariner.io/gateway=true** in the cluster.

5. Refer to Calico to configure Calico CNI by creating IPPools in the cluster.

1.1.2.3. Preparing Red Hat OpenShift Dedicated to deploy Submariner

Red Hat OpenShift Dedicated supports clusters that were provisioned by AWS and Google Cloud Platform.

1.1.2.3.1. Preparing Red Hat OpenShift Dedicated to deploy Submariner on AWS

To configure the AWS clusters on Red Hat OpenShift Dedicated, complete the following steps:

1. Submit a support ticket to the Red Hat OpenShift Hosted SRE Support team to grant **cluster-admin** group access to the Red Hat OpenShift Dedicated cluster. The default access of **dedicated-admin** does not have the permission that is required the create a **MachineSet**.

2. After the group is created, add the user name to the **cluster-admin** group that you created by completing the steps in Granting the cluster-admin role to users in the Red Hat OpenShift Dedicated documentation.

3. Configure the credentials of the user **osdCcsAdmin**, so you can use that as a service account.

4. Import your cluster to Red Hat Advanced Cluster Management, and follow the instructions in Deploying Submariner with the console .

1.1.2.3.2. Preparing Red Hat OpenShift Dedicated to deploy Submariner on Google Cloud Platform

To configure the Google Cloud Platform clusters on Red Hat OpenShift Dedicated, complete the following steps:

1. Configure a service account named **osd-ccs-admin** that you can use to manage the deployment.
2. Import your cluster to Red Hat Advanced Cluster Management, and follow the instructions in Deploying Submariner with the console .

1.1.3. Deploying Submariner with the console

The **submariner-addon** component is a **Technology Preview** feature.

You can deploy Submariner on Red Hat OpenShift Container Platform managed clusters that are deployed on Amazon Web Services, Google Cloud Platform, and VMware vSphere by using the Red Hat Advanced Cluster Management for Kubernetes console. To deploy Submariner on other providers, follow the instructions in Deploying Submariner with APIs . 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 navigation menu, 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 to deploy Submariner.
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. Enter the following information in the *Install Submariner add-ons* editor:
    - **AWS Access Key ID** - This field is only visible when you import an AWS cluster.
    - **AWS Secret Access Key** - This field is only visible when you import an AWS cluster.

- **Google Cloud Platform service account JSON key** - This field is only visible when you import a Google Cloud Platform cluster.

- **Instance type** - The Amazon Web Services EC2 instance type of the gateway node that is created on the managed cluster. The default value is **c5d.large**. This field is only visible when your managed cluster environment is AWS.

- **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 firewalls.

- **Gateway count** - The number of worker nodes that are used to deploy the Submariner gateway component on your managed cluster. 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**.

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 how many worker nodes are labeled with the Submariner gateway label: **submariner.io/gateway=true** on the managed cluster.

Submariner is now deployed on the clusters.

1.1.4. Deploying Submariner with APIs

Before you deploy Submariner with Red Hat Advanced Cluster Management for Kubernetes, you must prepare the clusters on the hosting environment for the connection. Currently, you can use the **SubmarinerConfig** API to automatically prepare the clusters on Amazon Web Services, Google Cloud Platform and VMware vSphere. For other platforms, you need to prepare them manually, see Preparing the hosts to deploy Submariner for the steps.

1.1.4.1. Preparing the hosts to deploy Submariner

Before you deploy Submariner with Red Hat Advanced Cluster Management, you must prepare the clusters on the hosting environment for the connection. The requirements vary for different hosting environments, so follow the instructions for your hosting environment.

1.1.4.1.1. Preparing Amazon Web Services to deploy Submariner

You can use the **SubmarinerConfig** API to configure the AWS cluster to integrate with a Submariner deployment. Follow the steps that apply to your situation to prepare AWS to install Submariner:

1. If you did not create the managed cluster with Red Hat Advanced Cluster Management, you must manually create a secret on your hub cluster in the namespace of your managed cluster that contains your AWS credential secret. If you created the cluster with Red Hat Advanced Cluster Management, then skip to step 2.
   To create the secret, enter a command that contains information that is similar to the following example and YAML. Run the command to apply the file:

   ```
   export AWS_ACCESS_KEY_ID=<aws-access-key-id>
   export AWS_SECRET_ACCESS_KEY=<aws-secret-access-key>

   cat << EOF | oc apply -f -
   ```

   ```
   apiVersion: v1
   kind: Secret
   metadata:
     name: <managed-cluster-name>-aws-creds
     namespace: <managed-cluster-namespace>
   type: Opaque
   data:
     aws_access_key_id: $(echo -n ${AWS_ACCESS_KEY_ID} | base64 -w0)
     aws_secret_access_key: $(echo -n ${AWS_SECRET_ACCESS_KEY} | base64 -w0)
   EOF
   ```

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

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

   Replace **aws-access-key-id** with your AWS access key ID.

   Replace **aws-secret-access-key** with your AWS access key.

2. If you created the managed cluster with Red Hat Advanced Cluster Management, or after you create the secret in the previous step, prepare the cluster by entering a command that is similar to the following example and YAML. Run the command to apply the file:

   ```
   cat << EOF | oc apply -f -
   ```

   ```
   apiVersion: submarineraddon.open-cluster-management.io/v1alpha1
   kind: SubmarinerConfig
   metadata:
     name: submariner
     namespace: <managed-cluster-namespace>
   spec:
     gatewayConfig:
      aws:
        instanceType: c5d.large
     credentialsSecret:
      name: <managed-cluster-name>-aws-creds
   EOF
   ```

   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-aws-creds** is your AWS credential secret name, which you can find in the cluster namespace of your hub cluster.

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

This configuration automatically opens the Submariner required ports: network address translation - traversal (NATT) port (4500/UDP), virtual extensible LAN (VXLAN) port (4800/UCP), and Submariner metrics port (8080/TCP) on your AWS instance. It also creates one AWS instance as the Submariner gateway with the AWS instance type **c5d.large**.

- If you want to customize your NATT port, enter a command that contains information that is similar to the following example and YAML. Run the command to apply the file:

  ```
  cat << EOF | oc apply -f -
  ```

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

  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-aws-creds** is your AWS 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.

- If you want to customize the AWS instance type of your gateway node, enter a command that contains information that is similar to the following example. Run the command to apply the file:

  ```
  cat << EOF | oc apply -f -
  ```

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

  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

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

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.

- If you want to customize the number of your gateway nodes, enter a command that contains information that is similar to the following example. Run the command to apply the file:

```
cat << EOF | oc apply -f -
```

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

+ 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-aws-creds** is your AWS 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.

1.1.4.1.2. Preparing Google Cloud Platform to deploy Submariner

You can use the **SubmarinerConfig** API to configure the Google Cloud Platform cluster to integrate with a Submariner deployment. Follow the steps that apply to your situation to prepare Google Cloud Platform to install Submariner:

1. If you did not create the managed cluster with Red Hat Advanced Cluster Management, you must manually create a secret on your hub cluster in the namespace of your managed cluster that contains your Google Cloud Platform credential secret. If you created the cluster with Red Hat Advanced Cluster Management, then skip to step 2.
   To create the secret, enter a command that contains information that is similar to the following example. Run the command to apply the file. Run the command to apply the file:

```
cat << EOF | oc apply -f -
```

```
apiVersion: v1
kind: Secret
metadata:
   name: <managed-cluster-name>-gcp-creds
```

```
  namespace: <managed-cluster-namespace>
type: Opaque
data:
  osServiceAccount.json: <gcp-os-service-account-json-file-content>
EOF
```

Replace **managed-cluster-name** with the name of your managed cluster. The value of **managed-cluster-name-aws-creds** is your Google Cloud Platform 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 **gcp-os-service-account-json-file-content** with the contents of your encoded Google Cloud Platform **osServiceAccount.json: $(base64 -w0)** file.

2. If you created the managed cluster with Red Hat Advanced Cluster Management, or you have already created the secret in the previous step, prepare the cluster by entering a command that is similar to the following example. Run the command to apply the file. Run the command to apply the file:

```
cat << EOF | oc apply -f -
```

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

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.

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

This configuration automatically opens the Submariner required ports: network address translation - traversal (NATT) port (4500/UDP), virtual extensible LAN (VXLAN) port (4800/UCP), and Submariner metrics port (8080/TCP) on your Google Cloud Platform instance. It also labels one worker node as the Submariner gateway and enables the public IP address of this node in your Google Cloud Platform cluster.

- If you want to customize your NATT port, enter a command that contains information that is similar to the following example. Run the command to apply the file. Run the command to apply the file:

```
cat << EOF | oc apply -f -
```

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

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.

- If you want to customize the number of your gateway nodes, enter a command that contains information that is similar to the following example. Run the command to apply the file. Run the command to apply the file:

```
cat << EOF | oc apply -f -
```

+

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

+ 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-aws-creds** is your Google Cloud Platform 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.

1.1.4.1.3. Preparing to deploy Submariner on VMware vSphere

Submariner uses IPsec to establish the secure tunnels between the clusters on the gateway nodes. You can use the default port or specify a custom port. When you run this procedure without specifying an IPsec NATT port, the default port is automatically used for the communication. The default port is 4500/UDP.

Submariner uses virtual extensible LAN (VXLAN) to encapsulate traffic when it moves from the worker and master nodes to the gateway nodes. The VXLAN port cannot be customized, and is always port 4800/UDP.

Submariner uses 8080/TCP to send its metrics information among nodes in the cluster, this port cannot be customized.

The following ports must be opened by your VMWare vSphere administrator before you can enable Submariner:

Table 1.2. VMware vSphere and Submariner ports

Name	Default value	Customizable
IPsec NATT	4500/UDP	Yes
VXLAN	4800/UDP	No
Submariner metrics	8080/TCP	No

To prepare VMware vSphere clusters for deploying Submariner, complete the following steps:

1. Ensure that the IPsec NATT, VXLAN, and metrics ports are open.

2. Enter a command that contains information that is similar to the following example to apply the YAML sample. Run the command to apply the file:

   ```
   cat << EOF | oc apply -f -
   ```

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

   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.

   This configuration uses the default network address translation - traversal (NATT) port (4500/UDP) for your Submariner and one worker node is labeled as the Submariner gateway on your vSphere cluster.

   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 of 4500/UDP is automatically used for the communication.

   - If you want to customize your NATT port, enter a command that contains information that is similar to the following example. Run the command to apply the file:

     ```
     cat << EOF | oc apply -f -
     ```

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

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

- If you want to customize the number of your gateway nodes, enter a command that contains information that is similar to the following example. Run the command to apply the file:

  ```
  cat << EOF | oc apply -f -
  ```

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

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

1.1.4.2. Deploy Submariner with the ManagedClusterAddOn API

To deploy Submariner by using the **ManagedClusterAddOn** API, complete the following steps:

1. Create a **ManagedClusterSet** on the hub cluster by using the instructions provided in Creating and managing ManagedClusterSets. Your entry for the **ManagedClusterSet** should resemble the following content:

   ```
   apiVersion: cluster.open-cluster-management.io/v1beta1
   kind: ManagedClusterSet
   metadata:
     name: <managed-cluster-set-name>
   ```

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

   Note: The maximum length of the name of the Kubernetes namespace is 63 characters, so the maximum length of the **<managed-cluster-set-name>** is 56 characters. If the length of **<managed-cluster-set-name>** exceeds 56, the **<managed-cluster-set-name>** is truncated 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. Add one managed cluster to the **ManagedClusterSet** by entering 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.

3. Deploy Submariner on the managed cluster by entering the following command:

```
cat << EOF | oc apply -f -
```

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

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**.

4. Repeat steps 2 and 3 for all of the managed clusters that you want to enable Submariner.

5. After Submariner is deployed on the managed cluster, you can verify the Submariner deployment status by checking the status of submariner **ManagedClusterAddOn** by entering 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.

1.1.4.3. Customizing Submariner deployments

You can customize some of the settings of your Submariner deployments.

1.1.4.3.1. 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.

1.1.5. Enabling service discovery for Submariner

The **submariner-addon** component is a **technology preview** feature.

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 **ManagedClusterSet**. To make a service from a cluster visible and discoverable to other clusters in the **ManagedClusterSet**, you must create a **ServiceExport** object. After a service is exported with a **ServiceExport** object, you can access the 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 that resembles the following content in the YAML file:

```
apiVersion: multicluster.x-k8s.io/v1alpha1
kind: ServiceExport
metadata:
  name: <service-name>
  namespace: <service-namespace>
```

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

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.