IBM Hybrid Cloud Mesh

Tables of Contents

Release notes

What's new

IBM Hybrid Cloud Mesh (Mesh) introduces the following features and functionality. For more overview information about Mesh, see Overview.

- Support for Red Hat® Service Interconnect

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

See Working with Red Hat® Service Interconnect.

- Enhanced Mesh console functions:
  - Auto-deploy a gateway
  - Add roles and permissions
  - Create a role with permissions
  - Add a user identity
  - View additional cloud details
- Added usage notes for Mac OSX arm64 download and install process.

- Global load balancing between multiple service deployments.

Known issues and limitations

IBM strives to keep issues in IBM Hybrid Cloud Mesh (Mesh) to a minimum, but occasionally they can't be avoided. Learn about the issues and limitations in this version. IBM is aware of these issues and is working hard to address them as quickly as possible.

Known issues

- If an application or service deployment moves from one Kubernetes worker node to another, any Mesh connection policies it was part of will no longer provide connectivity for it.
- When the `From` or `To` subject of a connection policy is replaced with another application or service, the connection policy is not properly updated. To work around this issue, delete the policy and create a new one.
- Networking on EKS fails under multi-worker nodes.

- Networking on IBM Kubernetes Service or Red Hat OpenShift Kubernetes Service (ROKS) fails under certain conditions in the Multi-node Multi-cluster scenario.

Mesh console

- Some specific console page hyperlinks open the Mesh landing page instead of the specific page.
- When associating infrastructure resources with a resource group, and the user wants to use the resource group for RBAC control, all the infrastructure resources with a cloud must be in the same resource group as the cloud.
- For RBAC, giving access to a parent resource does not propagate the permissions to the child resources. For more information, see Managing user roles, permissions, resource groups, and secrets.

Limitations

- You cannot use internal and external secrets at the same time
- Two clusters in the same Virtual Private Cloud (VPC) are not supported
- The procedure for deploying a gateway is currently only supported for AWS EKS.
- Only private `LoadBalancer` is supported in Kubernetes.
- You cannot connect a gateway to more than one deployment environment.

Conventions used in this document

This document uses content conventions to convey specific meaning.

Command conventions

- `< >` - Replace the variable content shown inside `< >` with values specific to your needs. Do not include the `< >` characters in the command. (As an alternative, this is sometimes shown as words in all capital letters.) Examples:

```
palmctl get application <application-id>
palmctl get application APPLICATION-ID
```

- `[ ]` - The content inside `[ ]` is optional. Example:

```
palmctl get applications [-h]
```

- `{ }` - The content inside `{ }` contains two or more choices separated by `|`. Choose one of them. (As an alternative, `( )` is sometimes used.) Example:

```
palmctl get policy {--id <policy-id> | --name <policy-name>}
```

Literal strings

- In instructions for using the Mesh console, words that are shown as **bold** text are literal strings that you will see in the console.
- In instructions for using the CLI, words that are shown as `monospace font` are literal strings that you should enter verbatim.

Supported platforms

IBM Hybrid Cloud Mesh (Mesh) supports the following platforms.

Mesh Edge gateway supported environment

The Mesh edge gateway supports the following deployment environments:

- AWS
  - Elastic Kubernetes Service (EKS)
- IBM Cloud®
  - IBM Cloud Kubernetes Service (IKS)
  - Red Hat® OpenShift® Kubernetes Service (ROKS)
- VMWare vSphere (on-prem)
  - Red Hat OpenShift Container Platform (OCP)

Supported Clouds for Infrastructure Auto-discovery

Supports the discovery of Locations, Clusters and VPCs.

- AWS
- IBM Cloud

Supported operating systems

The command-line interface (CLI) runs on the following operating systems:

- Red Hat Enterprise Linux® (RHEL): version 8.x
- Ubuntu: versions 20.x, 22.x
- Mac OSX: versions Big Sur and later
  - amd64
  - arm64
- Windows operating system: versions Windows 10 Pro and Windows 11 Pro

Supported browsers

Mesh supports the following browsers:

- Chrome: Recent versions
- Firefox: Recent versions

Overview of IBM Hybrid Cloud Mesh

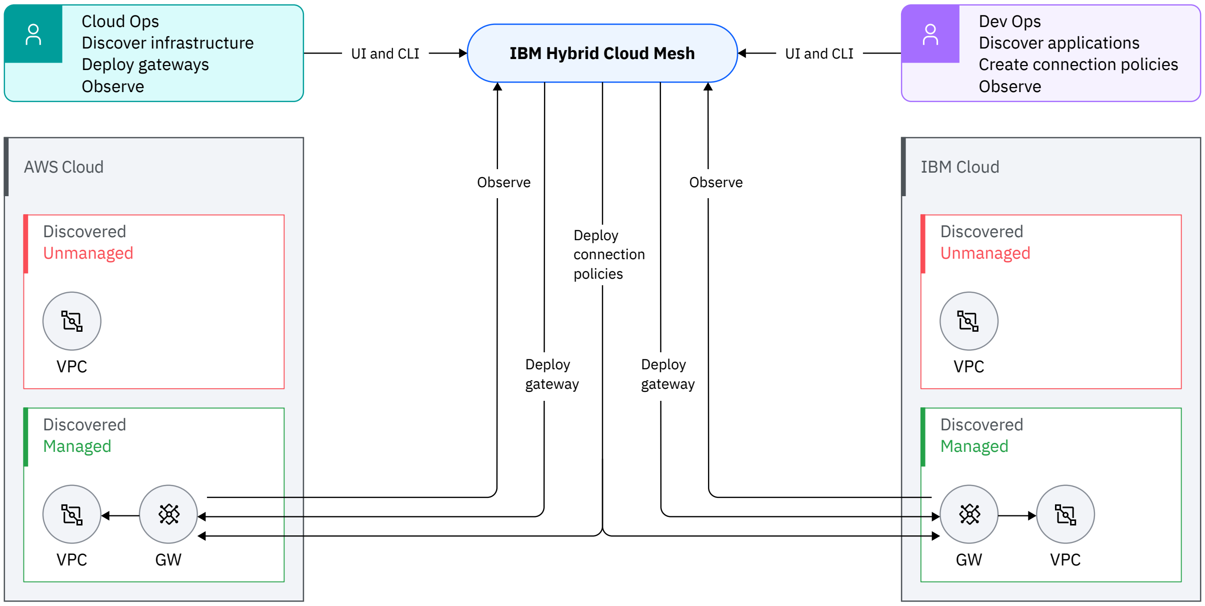
IBM Hybrid Cloud Mesh (Mesh) is a multi-cloud, multicluster, application-centric, networking solution. It enables enterprises to use simple, scalable, seamless, and secure hybrid multi-cloud connectivity. This software as a service (SaaS) solution delivers any service, anytime, anywhere, enabling application-centric networks by intelligently inferring network requirements from business intent. It aligns the networking operations, security operations, and DevOps across heterogeneous cloud environments.

Mesh automatically configures a software-defined network for the application's microservices, which are distributed among multiple clouds in an abstract manner. Mesh is an overlay network that eliminates the need for any reconfiguration of the underlying networks; for example, networks supporting the Amazon Cloud, Microsoft Azure, Google Cloud, and so on.

Mesh provides the following value to businesses:

- Improved business agility: Mesh improves business agility by enabling clients to deploy new applications and services faster.
- Enhanced performance and response time: Mesh can improve performance and user experience response time.
- Optimized security and cost: Mesh improves security by reducing the attack surface and lowers public cloud costs.
- Better visibility and seamless operation: Mesh gives networking and security professionals better visibility into their enterprise network security along with network and system performance, which provides better recommendations for improvement. At the same time, it saves DevOps and application developer professionals from the burden of juggling application connections across heterogeneous networks and cloud providers.

Figure 1 shows the relationship between Mesh and your applications and cloud components:



Functional overview

Adopting large numbers of multi-cloud applications where workloads are distributed across public clouds, edge devices, and on-premises data centers can cause unresponsive networks in Enterprise systems. The Mesh SaaS-based solution meets this challenge by delivering software that enables simple, scalable, seamless, and secure hybrid multi-cloud connectivity.

Mesh includes the following features:

Infrastructure Discovery: Creates an inventory of an enterprise's multi-cloud deployment infrastructure, which enables Mesh to understand the scope and breadth of the enterprise network. The results of this discovery provide enterprise CloudOps teams with full visibility into their mutli-cloud infrastructure. This feature requires credentials that can access enterprise cloud accounts and interrogate the cloud's API for assets. Periodic infrastructure discovery ensures that Mesh has the most current model of the enterprise infrastructure. Mesh uses infrastructure models to correlate applications and services with their supporting infrastructure. Examples of infrastructure include Virtual Private Cloud (VPC), Kubernetes clusters, Security Groups, and so on.

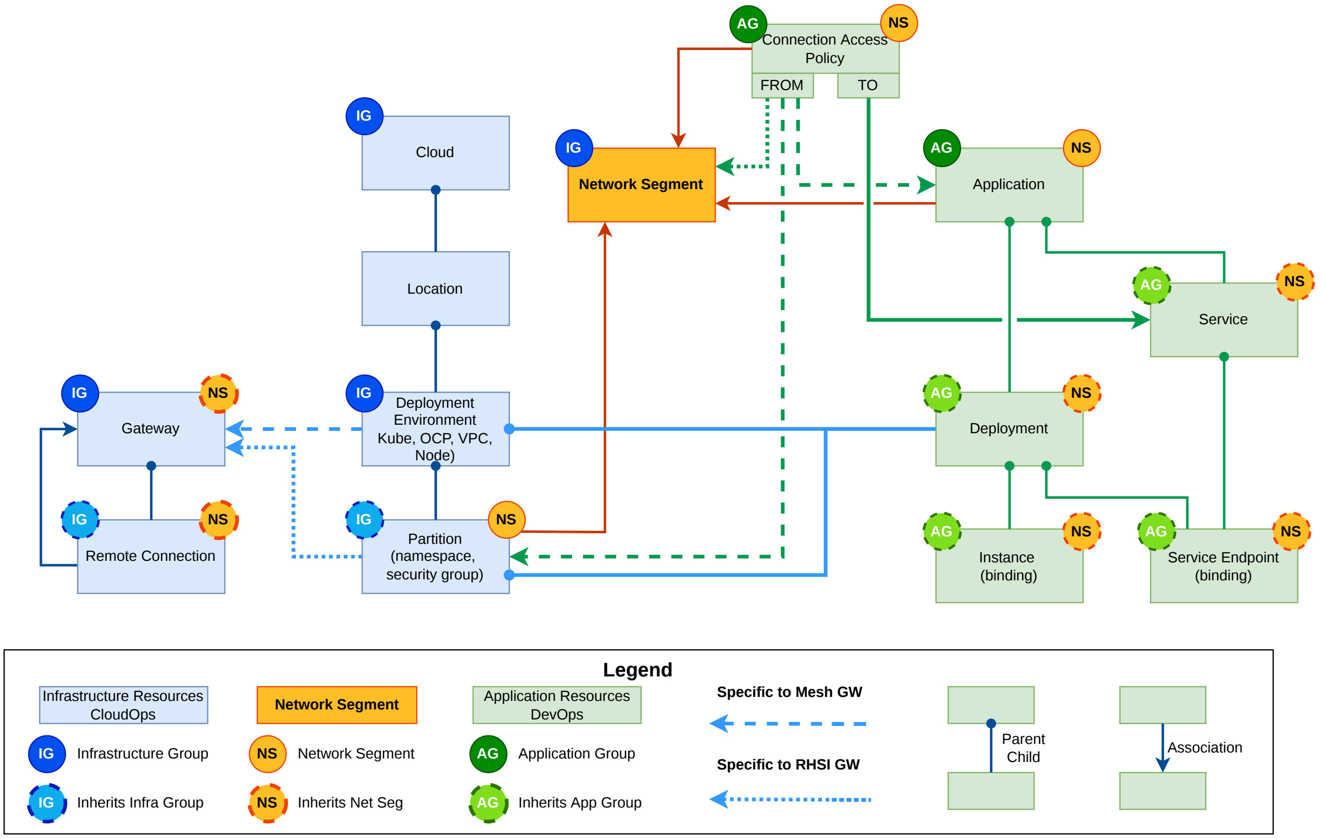
Application and Service Discovery: Creates an inventory of an enterprise's points of connectivity, which enables DevOps-driven policy intents to describe application and service connectivity. Applications and services are deployed by DevOps on infrastructure that is discovered by Infrastructure Discovery. DevOps has the freedom to deploy new versions or to move deployments from one part of the infrastructure to another. Mesh uses Application and service discovery to ensure that it is aware of applications and services as they migrate throughout the infrastructure. Applications and services are the main points of connectivity in Mesh. For example, an application in the **`Store`** Kubernetes namespace that needs connectivity to a service in the **`Inventory`** Kubernetes namespace requires that Mesh is aware of the deployment location of both the application and service, even as their deployment location changes over time.

Connectivity Management: Supports DevOps focused policy authoring to support connecting applications and services. Because Mesh is aware of applications and services that need to be connected and their deployment locations, DevOps can write simple policies expressing the intent to connect **`Store`** and **`Inventory`** wherever they are currently being deployed. This decouples policy authoring from the deployment mechanics of applications and services. It also enables a separation of concerns between the nature of the connection from the specifics of the endpoints being connected. The Mesh software gateways are responsible for enacting and enforcing the connection policies.

Network Topology: Provides an overlay network that is managed by software gateways that are automatically deployed to manage application and service endpoints. The software gateways are deployed like appliances in their own VM, close to the application and service deployments. The software gateways provide an overlay addressing scheme that ensures packets entering the overlay are delivered to the correct service instances. The topology views provide visibility to CloudOps and DevOps, enabling collaboration between teams. The topology views include metrics that describe network utilization over time.

Data model

The following illustration shows the relationship between all of the IBM Hybrid Cloud Mesh resources:



Regulatory compliance

IBM Hybrid Cloud Mesh (Mesh) complies with the following security and privacy assessments.

Customers are responsible for ensuring their own readiness for the laws and regulations that apply to them.

Customers are responsible for identifying and interpreting any relevant laws and regulations that might affect their users. Customers are also responsible for any actions their users might need to take to comply with these laws and regulations.

For more information, see the following topics:

- Considerations for GDPR Readiness
- Security and Privacy by Design (SPbD)

Considerations for GDPR Readiness

This document is intended to help you prepare for General Data Protection Regulation (GDPR) readiness. It provides IBM Hybrid Cloud Mesh Mesh feature information that you can configure, and aspects of the product's use to consider when you are preparing your organization for GDPR. This information is not an exhaustive list. Clients can choose and configure features in many ways and use the product in many behaviors and with third-party applications and systems.

Notice

Clients are responsible for ensuring their own compliance with various laws and regulations, including the European Union General Data Protection Regulation. Clients are solely responsible for obtaining the advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that might affect the client's business and any actions the clients might need to take to comply with such laws and regulations.

The products, services, and other capabilities that are described here are not suitable for all client situations and might restrict availability. IBM® does not provide legal, accounting, or auditing advice or represent or warrant that its services or products ensure that clients are in compliance with any law or regulation.

GDPR

GDPR was adopted by the European Union (EU) and applies from 25 May 2018.

Why is GDPR important?

GDPR establishes a stronger data protection regulatory framework for the processing of the personal data of individuals. GDPR brings:

- New and enhanced rights for individuals
- Widened definition of personal data
- New obligations for companies and organizations that are handling personal data
- Significant financial penalties for noncompliance
- Compulsory data breach notification

IBM established a global readiness program that is tasked with preparing IBM's internal processes and commercial offerings for compliance with the GDPR.

More information

- EU GDPR Information Portal

- GDPR

Product Configuration – considerations for GDPR Readiness

The following sections describe aspects of Mesh and provide information on capabilities to help clients with GDPR requirements.

Data Life Cycle

IBM Hybrid Cloud Mesh Mesh is a multicloud, multicluster, application-centric, networking solution. It enables enterprises to use simple, scalable, seamless, and secure hybrid multicloud connectivity. It enables application-centric networks by intelligently inferring network requirements from business intent. It aligns networking operations, security operations, and DevOps across heterogeneous cloud environments.

Mesh automatically configures a software-defined network for the application's microservices, distributed among multiple clouds in an abstract manner. Mesh is an overlay network that eliminates the need for any reconfiguration of the underlying networks.

Types of data stored in Mesh

Mesh stores only one kind of data that might be considered personal data:

- User emails

Information about how this data can be accessed and deleted is described in later sections of this document.

The user's email is their IBMid, which you can see in My IBM. The IBMid is used for user authentication and audit logs in Mesh. Only customers and IBM support can access the audit logs.

How to view personal data

To view the stored email, use the following command:

```
palmctl get identities
```

How to delete personal data

To remove the email, use the following command:

```
palmctl delete identity
```

Note: To remove all the email addresses for a customer or tenant, request removal by contacting IBM support.

Warning: If all the email addresses are deleted, the customer identity is erased and they can no longer use Mesh.

Legal Basis

For more information about the legal basis for the lawful handling of your personal data, see IBM Privacy Statement.

Security and Privacy by Design (SPbD)

Security and Privacy by Design (SPbD) at IBM® is an agile set of focused security and privacy practices, including threat models, privacy assessments, security testing, and vulnerability management.

IBM developed a set of SPbD processes and tools that are used by all of its business units. For more information about the IBM Secure Engineering Framework (SEF) and SPbD, see the IBM Redbooks® Security in Development - The IBM Secure Engineering Framework (PDF).

Getting started

New to IBM Hybrid Cloud Mesh (Mesh)? The following resources can help you get familiar with the platform, set up your Mesh console, and get productive.

Before you begin

- Create an IBMid and subscribe to Mesh at My IBM. You can log in to the Mesh console to verify your user ID.

Resources are not shown in the console until you complete the getting started steps.

- By default, the secrets manager strategy is set to use the internal Mesh secrets manager. You can use the CLI to set the secrets manager strategy to `external` to use an externally provided secrets manager. For more information about how to configure the secrets manager, see Secrets manager.

Getting the secrets

Get the IBM Cloud® and AWS secrets that are used to discover the cloud infrastructure. For more information about how to get the cloud secrets, see Getting cloud Secrets.

Discovering the cloud infrastructure

- Add cloud credentials to the Mesh instance of IBM Cloud secrets manager. The credentials are used to discover the cloud infrastructure.
    - **IBM Cloud discovery**: Add the IBM Cloud secret to the secrets manager.
    - **AWS discovery**: Add the AWS secret to the secrets manager.
- Register the cloud resource, and then display the discovered infrastructure on the cloud details page.

For detailed steps, see Discovering cloud infrastructure.

Discovering cluster namespaces and autodeploying gateways

- Find the Deployment environment (cluster) on the cloud details page.
- Toggle **Managed** to **On** for your parent location and the Deployment environment.
- Toggle **Autodiscover** to **On** on the Deployment environment to enable auto-discovery of the namespaces.

The instance starts to configure the deployment environment, and autodeploying the gateways. It might take some time to deploy the gateway and autodiscover the cluster namespaces. For detailed steps about how to discover the cluster namespaces and autodeploy the gateways, see Discovering cluster namespaces and applications.

Discovering applications

- Expand the Deployment environment on the cloud details page to see the discovered namespaces.
- Toggle **Managed** to **On** for the namespace where you deployed the applications you want to discover.
- Toggle **Autodiscover** to **On** to autodiscover all applications.

For detailed steps about how to discover the applications, see Discovering cluster namespaces and applications.

Creating connection policies

You can create connection policies by using the Mesh console or the CLI. To create connection policies, you must have applications that are deployed in multiple environments with gateways.

Creating policies by using the Mesh console

- **Application to service policy**: Enables connectivity from any deployment of the application to any deployment of the service. You can create a policy in the Mesh console, and specify an application on the **From** side and a Service on the **To** side.
- **Partition to service policy**: Enables connectivity from any application that is deployed in the partition to any deployment of the service. You can create a policy in the Mesh console, and specify a partition (namespace) on the **From** side and a Service on the **To** side.

Creating policies by using the CLI

You can create connection policies by using the CLI, and automate the creation of connection policies as part of your CI/CD pipeline.

For detailed steps about how to create connection policies, see Managing application connections using policies.

What to do next

To learn more about managing policies and resources, and viewing topology information, see Using the Mesh console.

Secrets manager

When you create a secret in IBM Hybrid Cloud Mesh, through IBM® Hybrid Cloud Mesh console, the secret is stored in the IBM Hybrid Cloud Mesh (Mesh) secrets manager, which is an internal secrets manager by default. However, you can choose to store secrets in an external secrets manager, and update the secrets manager type to external or internal through the CLI. For more information, see Managing secrets.

Make the Getting started steps easier by allowing IBM Hybrid Cloud Mesh (Mesh) to discover your cloud infrastructure and applications. You can allow the Mesh to discover your cloud infrastructure and applications by providing some low-privileged credentials to Mesh by putting them in your secrets manager instance. The secrets manager for internal type secret is IBM Hybrid Cloud Mesh (Mesh), and the secrets manager for external type secret is IBM® secrets manager.

Using an external secrets manager (optional)

1. Provision an instance of the IBM secrets manager:
   1. Go to the IBM Cloud® Console and create a cloud account, if necessary.
   2. On the main console page, click **Create resource**, and search for `Secrets Manager`.
   3. Select a plan, enter the configuration details, accept the terms, and click `Create`.
2. Create an API key for the IBM Cloud service ID that has read-access to the secrets that you provide to Mesh.
3. Note the value of the API key and substitute it for `<your-secrets-manager-api-key>` when you configure Mesh discovery. For more information, see Discovering cloud infrastructure.

Getting cloud secrets

IBM Hybrid Cloud Mesh (Mesh) uses IBM Cloud® and AWS API keys to discover the cloud infrastructure.

IBM Cloud API key

The following IBM® permissions are required for infrastructure discovery, gateway deployment, and interacting with IBM Kubernetes Service or Red Hat® OpenShift® Kubernetes Service (ROKS) clusters:

- **VPC Infrastructure services** (`is`): Reader, Writer, IP Spoofing Operator, Viewer, Operator, Editor, Manager
- **Default resource group**: Viewer
- **Kubernetes** (`containers-kubernetes`): Manager, Operator, Viewer, Editor, Administrator
- **IAM Identity** (`iam-identity`): Administrator

If IBM secrets manager is used to provide secrets, make sure `SecretReader` or `secret-group` is used for the secret in the IBM secrets manager instance. IBM Cloud users can create an access group with the specified permissions and generate a serviceID with API key to use as a secret.

AWS API key

The following AWS permissions are required for infrastructure discovery, gateway deployment, and interacting with EKS clusters:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Action": [
                                "ec2:AllocateAddress",
                                "ec2:AssociateAddress",
                                "ec2:AssociateRouteTable",
                                "ec2:AttachInternetGateway",
                                "ec2:AuthorizeSecurityGroupEgress",
                                "ec2:AuthorizeSecurityGroupIngress",
                                "ec2:CreateInternetGateway",
                                "ec2:CreateKeyPair",
                                "ec2:CreateNetworkInterface",
                                "ec2:CreateRoute",
                                "ec2:CreateRouteTable",
                                "ec2:CreateSecurityGroup",
                                "ec2:CreateSubnet",
                                "ec2:CreateTags",
                                "ec2:CreateVpc",
                                "ec2:DeleteInternetGateway",
                                "ec2:DeleteKeyPair",
                                "ec2:DeleteNetworkInterface",
                                "ec2:DeleteRoute",
                                "ec2:DeleteRouteTable",
                                "ec2:DeleteSecurityGroup",
                                "ec2:DeleteSubnet",
                                "ec2:DeleteVpc",
                                "ec2:DescribeAvailabilityZones",
                                "ec2:DescribeNetworkInterfaces",
                                "ec2:DescribeSubnets",
                                "ec2:DetachInternetGateway",
                                "ec2:DisassociateAddress",
                                "ec2:DisassociateRouteTable",
                                "ec2:ImportKeyPair",
                                "ec2:ModifyImageAttribute",
                                "ec2:ModifyInstanceAttribute",
                                "ec2:ModifyNetworkInterfaceAttribute",
                                "ec2:ModifySubnetAttribute",
                                "ec2:ReleaseAddress",
                                "ec2:ReplaceRoute",
                                "ec2:ReplaceRouteTableAssociation",
                                "ec2:RunInstances",
                                "ec2:TerminateInstances",
                                "route53:ChangeResourceRecordSets",
                                "route53:CreateHostedZone",
                                "route53:DeleteHostedZone",
                                "route53:ListHostedZonesByVPC",
                                "route53:ListResourceRecordSets",
                                
"route53resolver:AssociateResolverEndpointIpAddress",
                                "route53resolver:AssociateResolverRule",
                                "route53resolver:CreateResolverEndpoint",
                                "route53resolver:CreateResolverRule",
                                "route53resolver:DeleteResolverEndpoint",
                                "route53resolver:DeleteResolverRule",
                                
"route53resolver:DisassociateResolverEndpointIpAddress",
                                "route53resolver:DisassociateResolverRule",
                                "route53resolver:Get*",
                                "route53resolver:GetResolverEndpoint",
                                "route53resolver:GetResolverRule",
                                "route53resolver:GetResolverRuleAssociation",
                                "route53resolver:UpdateResolverEndpoint",
                                "route53resolver:UpdateResolverRule"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                },
                {
                        "Action": [
                                "ec2:*",
                                "eks:*",
                                "elasticloadbalancing:*",
                                "kms:*",
                                "logs:*",
                                "s3:*",
                                "ssm:*"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                },
                {
                        "Action": [
                                "kms:CancelKeyDeletion",
                                "kms:CreateAlias",
                                "kms:CreateKey",
                                "kms:CreateGrant",
                                "kms:Decrypt",
                                "kms:DescribeKey",
                                "kms:DisableKey",
                                "kms:EnableKey",
                                "kms:Encrypt",
                                "kms:GenerateDataKey*",
                                "kms:ListAliases",
                                "kms:ListKeys",
                                "kms:ListResourceTags",
                                "kms:PutKeyPolicy",
                                "kms:RetireGrant",
                                "kms:RevokeGrant",
                                "kms:ReEncrypt*",
                                "kms:ScheduleKeyDeletion",
                                "kms:TagResource",
                                "kms:UntagResource",
                                "kms:UpdateAlias"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                },
                {
                        "Action": [
                                "sts:AssumeRole",
                                "sts:AssumeRoleWithWebIdentity",
                                "sts:GetCallerIdentity"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                },
                {
                        "Action": [
                                "ec2:Describe*",
                                "ec2:List*",
                                "eks:Describe*",
                                "eks:List*",
                                "route53:List*",
                                "route53resolver:List*"
                        ],
                        "Effect": "Allow",
                        "Resource": "*"
                }
        ]
}
```

Discovering cloud infrastructure

To connect applications, IBM Hybrid Cloud Mesh (Mesh) must know about the infrastructure that those applications run on. Inform Mesh about the infrastructure in one of two ways:

1. Manually register the infrastructure with Mesh. See Registering infrastructure resources.
2. Have Mesh discover the cloud infrastructure. This process is described in this section.

Prerequisites

To turn on autodiscover, you must first register a secret with Mesh. If you want to use the external secrets manager see configuring the external secrets manager

Autodiscovering AWS

Mesh enables you to autodiscover the cloud infrastructure through Mesh console. Follow this process to enable autodiscover for AWS:

Get the access key ID and secret access key

To autodiscover the AWS you need to get the access key ID and secret access key. Follow the steps below to create the access key ID and secret access key:

1. Go to your AWS account. Go to your profile, and click **Security credentials**.
2. Scroll down to **Access key** and click **Create access key**.
3. Choose third party service and give it a name.
4. Copy the Access key ID and Secret access key and save it on your local machine.

Add the access key ID and secret access key to the internal secret manager through Mesh console

While creating an internal secret through Mesh console add the access key ID and secret access key to the Hybrid Cloud Mesh Secrets Manager. Follow these steps to add or register secrets to Hybrid Cloud Mesh Secrets Manager through Mesh console.

1. Expand to **Manage** dropdown and click on **Secrets**.
2. Click on **Register secret**.
3. Provide a secret name. By default the secret is stored in Hybrid Cloud Mesh Secrets Manager which is the internal secret manager and Infrastruture group is default infrastructure group.
4. Choose the secret type as AWS.
5. Provide the access key ID and secret access key and click **Register**.

Enable autodiscover for AWS

Once the secrets are added to the secret manager follow these steps to autodiscover AWS through Mesh console :

1. Click **Register Cloud** on **Clouds** page.
2. Select a AWS from the provider list displayed by clicking on the option and click **Next**.
3. Provide the Name and Description. By default, the Infrastruture group will be Default Infrastructure Group.
4. Toggle the **Autodiscover** toggle button **On** and select a secret name from the drop-down list.
5. Click **Register**.

Alternatively, if you choose to use external secret manager, add the secret to IBM® Secret Manager that contains these key/value pairs. Once the IAM AccessKey is created that has access to `list` and `describe` EC2 instances, VPCs, and EKS clusters in your AWS account to discover the cloud infrastructures, then add the secret to IBM Secret Manager, get the secret path through IBM Cloud® Console and add it to Mesh console while registering a secret:

```
{
  "accessKeyId": "<your-AWS-account-IAM-access-key-ID>",
  "secretAccessKey": "<your-AWS-account-IAM-secret-access-key>"
}
```

- Set the cloud and secret types in variables for subsequent commands:

  ```
  CLOUD_TYPE=AWS
  SECRET_TYPE='cloud-aws'
  ```

Getting the secret path

For the external secret manager, get the path to it by following these steps and add it to Mesh console while registering a secret:

- In the IBM Secrets Manager console:
  - Click the secret just created.
  - In the `Details` side-panel, click the `Actions` menu, and click `Show snippet`.
  - Click `Curl` tab.

    - Copy the URL in the `curl` command.
- This URL is a combination of the Secrets Manager service API URL, the path to the secret, and the ID of the secret.

  Provide the copied path to the Mesh console when registering a secret to the external secret Manager.

  For {site.data.keyword.prod_short_name}}CLI, set it in the following variable:

  ~~~
  SECRET_URL=<URL-to-secret>
  ```**Note**: Getting the secret path process applicablies to both AWS and IBM cloud.
  ~~~

IBM Cloud Discovery

Get the API key

To autodiscover the IBM cloud you need to create an IBM API key for an IBM Cloud service ID that has read access to the secrets you will provide to Mesh.

Add the API key to the internal secret manager through Mesh console

While creating an internal secret through Mesh console add the API key to the Hybrid Cloud Mesh Secrets Manager. Follow these steps to add or register secrets to Hybrid Cloud Mesh Secrets Manager through Mesh console.

1. Expand to **Manage** dropdown and click on **Secrets**.
2. Click on **Register secret**.
3. Provide a secret name. By default, the secret is stored in Hybrid Cloud Mesh Secrets Manager which is internal secret manager and Infrastructure group is default infrastructure group.
4. Choose the secret type as IBM cloud.
5. Provide the API key and click **Register**.

Enable autodiscover for IBM cloud

Once the internal secret is added to the secret manager follow these steps to autodiscover IBM cloud through Mesh console :

1. Click **Register Cloud** on **Clouds** page.
2. Select a IBM from the provider list displayed by clicking on the option and click **Next**.
3. Provide the Name and Description. By default the Infrastructure group will be Default Infrastructure Group.
4. Toggle the **Autodiscover** toggle button **On** and select a secret name from the drop-down list.
5. Click **Register**.

If you choose to use external secret manager, add the secret to IBM Secret Manager that contains these key/value pairs, then add the secret to IBM Secret Manager, get the secret path through IBM Cloud Console and add it to Mesh console while registering a secret:

- Add a secret to secrets manager that contains this key/value pair:

  ```
  {
    "apikey": "<your-IBM-account-api-key>",
  }
  ```

Get the path as described previously and provide the copied path to the Mesh console when adding secret to the external secret Manager through Mesh.

Notes

- If you want a Mesh gateway to discover partitions (namespaces) and applications running in an IBM Cloud Kubernetes Service (IKS) cluster, then the secret provided must have `admin` access to the cluster, because Mesh modifies the IKS Container Network Interface (CNI) for reserving the IP addresses of the pods.
- Set the cloud and secret types in variables for subsequent commands:

  ```
  CLOUD_TYPE=IBM
  SECRET_TYPE='cloud-ibm'
  ```

- Currently, IBM Hybrid Cloud Mesh (Mesh) supports the discovery of Virtual Private Cloud (VPC) and VPC-based Kubernetes clusters. If you have any Kubernetes classic cluster then it will not be discovered.

Creating a reference to a secret in Mesh

Inform Mesh about the secret by creating a reference to it.

Example:

```
cat << EOM | palmctl create secret -f -
name: my-${CLOUD_TYPE}-cloud-secret
type: $SECRET_TYPE
backend: ibm-secrets-manager
path: $SECRET_URL
auth:
  api_key: <your-api-key-to-access-secrets-manager>
EOM
```

Notes about the secret fields:

- The `name` field value is used in the `credentials_key` field of other resources when creating other resources that reference this secret.
- Valid values for `type` are: `cloud-aws`, `cloud-ibm`, and `credentials-k8s`.

Registering the cloud resource

Before registering the cloud resource, decide if you want the discovered resources placed in the `Default_Infrastructure_Group` or in a different resource group:

- If you want the discovered resources placed in the `Default_Infrastructure_Group`:

  ```
  RESOURCE_GROUP_ID='default-infra'
  ```

- If you want the discover resources placed in a different resource group:
  - Follow Creating resource groups to create your resource group.
  - In the output of the create command, note the value of the `resource_id` field and set it in shell variable `RESOURCE_GROUP_ID`.

Register the cloud that Mesh should discover, turning on auto-discover and providing a secret.

Note: The following example uses the Mesh CLI to register the cloud. The cloud can also be registered using the Mesh console and specifying equivalent values.

Example:

```
cat << EOM | palmctl create cloud -f -
name: $CLOUD_TYPE
type: $CLOUD_TYPE
auto_discover: true
credentials_key: my-${CLOUD_TYPE}-cloud-secret
resource_group_id: $RESOURCE_GROUP_ID
EOM
```

Notes about the cloud fields:

- The value of the **`name`** field can be anything, but it must unique.
- The valid values for **`type`** are: **`IBM`**, **`AWS`**, **`other`**.

Note: Auto-discovery cannot be performed on clouds of type **`other`**.

Viewing infrastructure resources

When the cloud is registered (the resource created), Mesh automatically starts discovering the cloud components and creating corresponding resources in Mesh. You can monitor its progress by going to the Mesh console and refreshing the **`Locations`** list page and the **`Deployment environments`** list page.

See Viewing infrastructure resources for details.

Selecting the managed resources

Infrastructure discovery can discover more infrastructure than is relevant to the applications you want Mesh to connect to.

Therefore, Mesh sets the **`unmanaged`** field of all of the discovered resources to **`true`**. To select which resources should be managed, navigate to each resource in the Mesh console and change the setting to **`Managed`**. This sets the resource's **`unmanaged`** field to **`false`**. By default, only managed resources are displayed in the Mesh console. This lets you focus on the most important infrastructure resources.

Note: Click the **`Managed`** drop-down menu on any list page and select **`Unmanaged`** or **`All`** to view unmanaged resources in the Mesh console.

Discovering cluster namespaces and applications

A cluster is an example of a deployment environment, which is an infrastructure that can run applications. IBM Hybrid Cloud Mesh (Mesh) uses a gateway for each deployment environment to enable connections between an application that runs in one deployment environment to a service in another deployment environment. The deployment environments can be located in separate clouds or on-prem data centers.

Mesh uses one of the following methods to know which namespaces are in each cluster, and which applications and services are running in each namespace.

- Manually register the clusters and applications. For more information, see Registering infrastructure resources and Registering application resources.

- Mesh discovers the cluster namespaces and applications. If you deployed the gateway for a cluster by using the **`palmctl create gwdeployment`** command, then the cluster is configured to discover the namespaces and applications.

Map role-based access to the AWS EKS cluster

If you do not have admin privileges on the cluster and you are using the **mapRoles** configuration to provide cluster access for the user, then you must complete these steps. For more information, see Enabling access to your AWS EKS cluster.

To enable Mesh to discover the namespaces with your credentials, complete the following steps on the AWS EKS cluster:

1. Create an Identity and Access Management (IAM) role that is called **k8sadmin.mesh.ibm**. This role is used to access the cluster and must be configured as follows:

    1. The role must have admin privileges for the cluster because Mesh modifies the AWS EKS Container Network Interface (CNI).

    2. You must configure the role with the **AmazonEKSClusterPolicy** managed policy.

    3. You must create the role with the appropriate trust policies that define which AWS accounts can assume the role. For example:

        ```
        {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::{YourAWSAccountID}:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
        ```

2. Create an IAM user group that is called **k8sadminUserGroup.mesh.ibm**.

3. While creating the user group, create a policy that is attached to the group called **k8sadminPolicy.mesh.ibm**. The policy must allow the **AssumeRole** action on the **k8sadmin.mesh.ibm** role. For example:

    ```
    {
        "Version": "2012-10-17",
        "Statement": [
            {
                "Sid": "VisualEditor0",
                "Effect": "Allow",
                "Action": "sts:AssumeRole",
                "Resource": "arn:aws:iam::
    {YourAWSAccountID}:role/k8sadmin.mesh.ibm"
            }
        ]
    }
    ```

4. Add the users that need to access the AWS EKS cluster to the **k8sadminUserGroup.mesh.ibm** user group.

5. Add the **k8sadmin.mesh.ibm** role to the **aws-auth ConfigMap** of the AWS EKS cluster, as follows:

1. Edit the configmap:

   ```
   kubectl -n kube-system edit configmap aws-auth
   ```

2. Add the `k8sadmin.mesh.ibm` role:

   ```
   apiVersion: v1
   data:
   mapRoles: |
       - groups:
       - system:masters
       rolearn: arn:aws:iam::{YourAWSAccountID}:role/k8sadmin.mesh.ibm
       username: admin
   ```

Configure the automatic discovery of namespaces

To automatically discover the namespaces in your deployment environment, complete the following steps in the Mesh console:

1. From the navigation menu, click **Deployment environments** (⁘).
2. Click **View**, then select **All** from the drop-down list to view all the deployment environments.
3. Click the deployment environment where you want to automatically discover the namespaces, then toggle **Managed** to **On**.

   You must also set the location of your deployment environment to `Managed`.

4. Click the Edit icon ✎, and connect a gateway on the deployment environment.

   The automatic discovery turns on when you connect a gateway to the deployment environment.

5. From the drop-down list, select the name of the secret that is deployed to discover your deployment environment.
6. Click **Save changes**.

To see the discovered namespaces, go to your deployment environment, then refresh the list of the namespaces.

Configure the automatic discovery of applications in the namespace

To automatically discover the applications that are running in your namespaces, complete the following steps in the Mesh console:

1. From the deployment environment, click **View**, then select **All** from the drop-down list to view all the namespaces.
2. Click the namespace where you want Mesh to automatically discover the applications, then toggle **Managed** to **On**.
3. Click the Edit icon ✎, then toggle **Autodiscover** to **On** on the namespace to enable auto-discovery of the applications.
4. Click **Save changes**.

To see the discovered applications, go to your namespace, then refresh the list of the applications.

Using the Mesh CLI

Use the Mesh CLI to configure the automatic discovery of namespaces, and applications in the namespace.

Configure the automatic discovery of namespaces by using the CLI

Configure the automatic discovery of namespaces in the cluster by allowing the Mesh to manage the cluster. For example, run a command like this to allow the Mesh to manage the AWS EKS cluster that is called **`kube-1`** to enable auto-discovery of the namespaces in the AWS EKS cluster.

```
cat << EOM | palmctl patch cluster --cloud-name AWS --name kube-1 -f -
unmanaged: false
auto_discover: true
EOM
```

Configure the automatic discovery of applications in the namespace by using the CLI

Configure the automatic discovery of applications in the namespace by allowing the Mesh to manage the namespace. For example, run a command like this to allow the Mesh to manage the namespace that is called **`namespace-1`** to enable auto-discovery of the applications in the namespace.

```
cat << EOM | palmctl patch namespace --cloud-name AWS --cluster-name kube-1 --name
namespace-1 -f -
unmanaged: false
auto_discover: true
EOM
```

Managing application connections using policies

Connection policies are used to control access to a network. Policies define the types of connections that can be made and the encryption protocols used. They are used to protect a network from unauthorized access and to ensure that the data is transmitted encrypted. When applications cross multiple clouds, policies can allow connectivity between an application and a service or connect all applications in a deployment environment partition to a service. You can create connection policies using the Mesh console or the CLI and you can also automate the creation of connection policies as part of your CI/CD pipeline.

Prerequisites

- Register deployment environments using auto-discovery or manually
- Register applications using auto-discovery or manually.

Create connection policies

Connection policies instruct Mesh to enable connections between specific applications and services.

Create connection policies using the Mesh console

Use connection policies to easily control which applications can communicate with which services.

- Use the Mesh console connection policy page to create connection policies and view existing policies.

- You can create policies to connect a single application to a single service.
- You can encrypt the policy connections by checking **Use encrypted path**, while creating or editing a policy.
- By default policies belong to a default network segment. Policy names must be unique within the network segment. See Working with Red Hat® Service Interconnect to learn how to work with network segments and the RHSI integration tech preview.

Adjust the domain name used to connect to your services

Adjust your applications to connect to each service using the fully qualified domain name (FQDN) that was specified in the `name` field when the service was registered with Mesh. (This FQDN should end with `palmetto.ibm.com`.)

Create connection policies using CLI

Connection policies instruct IBM Hybrid Cloud Mesh (Mesh) to enable connections between specific applications and services.

Creating a connection policy between one application and a service

Example:

```
cat << EOM | palmctl create policy -f -
name: mypolicy
from:
  type: application
  application:
    application_name: <app1-name>
to:
  type: service
  service:
    application_name: <app2-name>
    service_name: <svc2-fqdn>
action: ALLOW   # this is currently the only valid value for action
EOM
```

Creating a connection policy between a partition and a service

A policy can enable the connection of every application in a partition (cluster namespace or VPC security group) to a service.

Example:

~~~
cat << EOM | palmctl create policy -f -
name: mypolicy2
from:
  type: deployment_env
  deployment_env:
    deployment_env_name: <dep-env-name>
    partition_name: <partition-name>
to:
  type: service
  service:
    application_name: <app2-name>
    service_name: <svc2-fqdn>
action: ALLOW   # this is currently the only valid value for action
EOM
```## Deploying a policy in the CI/CD pipeline
~~~

```
The following steps apply to the AXON-NET gateway with autodiscovery of
applications.

1. Deploy applications and services using the user pipeline.
2. Autodiscovery detects the applications and services and create Application,
Service, Application Deployment, Instance and Endpoint records.

**Note**: In the user pipeline you can [create an application to service policy]
(#creating-a-connection-policy-between-one-application-and-a-service) and [create
a partition to service policy](#creating-a-connection-policy-between-a-partition-
and-a-service).

## What's Next

[View the application real time network traffic](../ui/viewing-app-traffic.html)
using Grafana.
```

Viewing application network traffic

Connect your Grafana dashboard to IBM Hybrid Cloud Mesh (Mesh) by completing the following steps.

Prerequisites

- Install Grafana. For installation instructions, see Grafana installation.

Note: Grafana 9.2.x or later is required.

- Use the Grafana provisioning feature to configure your Grafana dashboard and data source. See Provisioning Grafana and Provisioning dashboards and data sources.

Running Grafana on Linux®

Complete the following steps:

1. Configure the following environment variables for Grafana: `MDM_API_KEY` and `MDM_SERVICE_URL`.

   `MDM_API_KEY` is used to authenticate the tenant in the data source. The API key is needed for any user that needs read access to all the Mesh resources.
   To generate your API key, see Managing API keys.

2. On Linux, Grafana uses a specific `EnvironmentFile`, which is set to `/etc/sysconfig/grafana-server`. Add the environment variables to this file. Use the following example settings as a guide:

   ```
   MDM_API_KEY=xxxxxxxxxxxxxx
   MDM_SERVICE_URL=https://app.hybridcloudmesh.ibm.com/api/v1/metrics
   ```

3. Place the following configuration files in the specified directories to provision Grafana with a data source and a dashboard:

   - To create a data source, place the data source configuration file in `/etc/grafana/provisioning/datasources/`.
   - To configure a dashboard, place the dashboard configuration file in `/etc/grafana/provisioning/dashboards/`.

    - To create a dashboard, place the dashboard definition file in `/var/lib/grafana/dashboards/`.
4. Restart the Grafana instance.

Running Grafana in Docker

Complete the following steps:

1. Export the following variables to the Docker container and include the configurations in the Grafana configuration for the container:

   ```
   MDM_API_KEY=xxxxxxxxxxxxxx
   MDM_SERVICE_URL=https://app.hybridcloudmesh.ibm.com/api/v1/metrics
   ```

   For information about mounting volumes, see Start a container with a volume.
   For information about passing environment variables to containers, see Pass environment variables to containers.

2. Use volume mounting to include the following configuration files in the container's Grafana configuration:

   - To create a data source, place the data source configuration file in `/etc/grafana/provisioning/datasources/`.
   - To configure the dashboard, place the dashboard configuration file in `/etc/grafana/provisioning/dashboards/`.
   - To create the dashboard, place the dashboard definition file in `/var/lib/grafana/dashboards/`.
3. Restart the Grafana instance

Logging in to Grafana

To log in to Grafana for the first time:

1. Open your web browser and go to the Grafana URL.
2. Enter **admin** as the username and password.
3. Click **Sign in**. When you are signed in, you are prompted to change the password.
4. Click **OK** and change your password.

To view the dashboard, hover over the **Dashboards** (squares) icon in the sidebar, and click **Manage**. The dashboard appears in the `General` folder.

Validating Grafana connectivity

Complete the following steps:

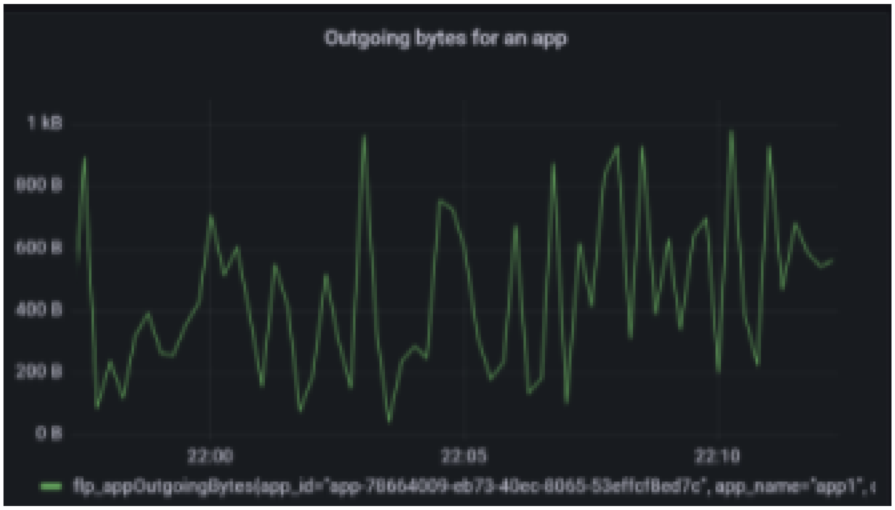
1. After you log in to Grafana and reset your password, hover over the **Configuration** icon in the left pane.
2. Click **Data sources** in the menu.
3. Click **Metrics datasource Manager** on the middle pane list.
4. In the HTTP section, ensure that the URL matches the `MDM_SERVICE_URL` variable.
5. Scroll to the bottom and click **Test**. If the URL is correct and the API key is authenticated, the following message is shown: "Data source is working".

Viewing application traffic through Grafana

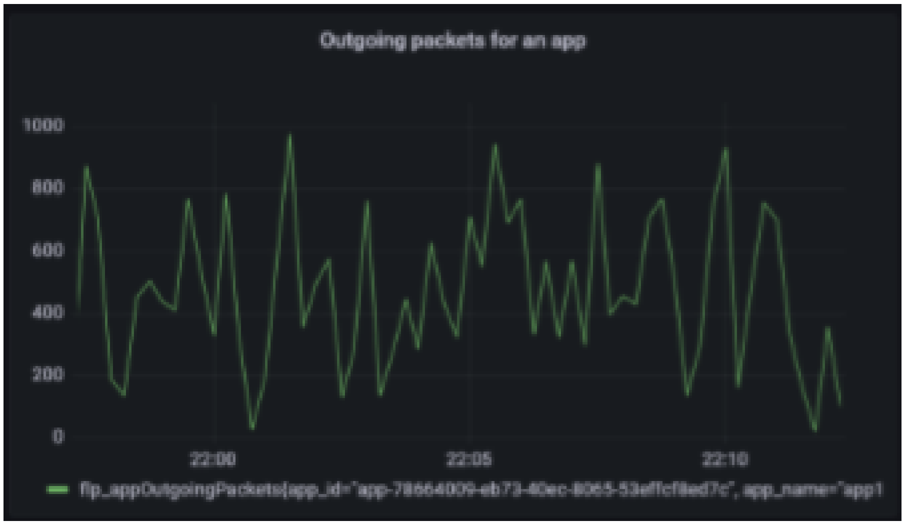
- To view the charts that display outgoing bytes or packets, select the application from the **App Name** drop-down.
- To view the charts that display incoming bytes or packets, select the service from the **Service Name** drop-down.

You can select multiple applications and services. The following charts display the real-time application traffic between the selected application and the service:

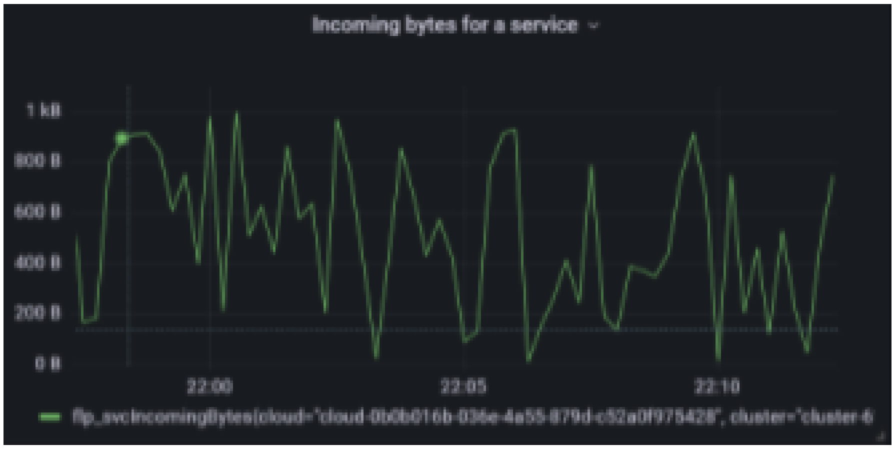
- Outgoing bytes for an app



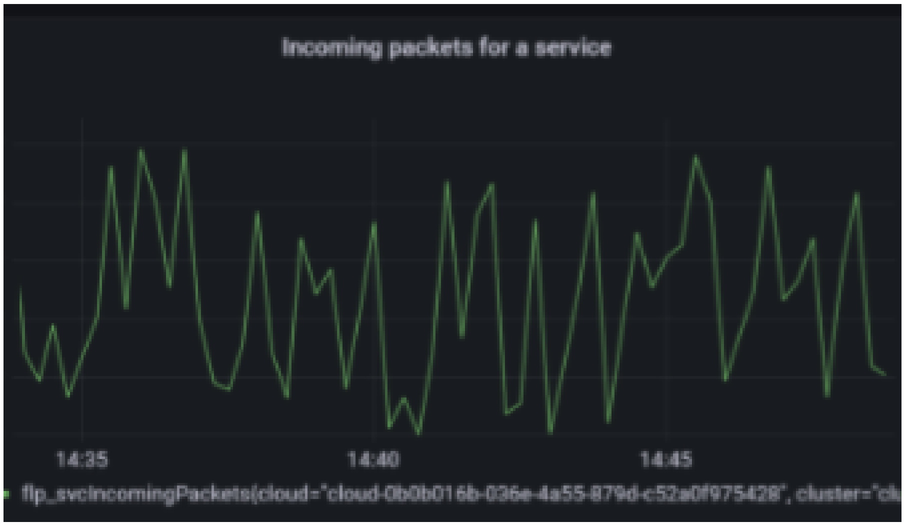
- Outgoing packets for an app



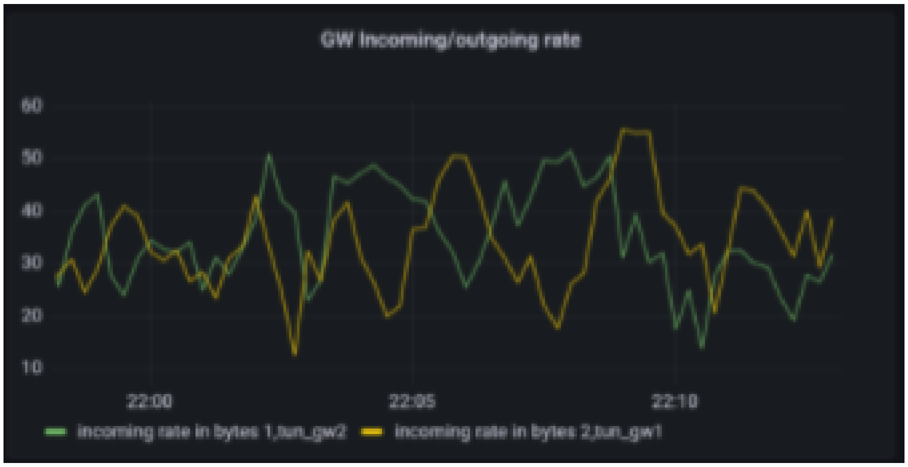
- Incoming bytes for a service


Incoming bytes for a service
1 kB
800 B
600 B
400 B
200 B
0 B
22:00
22:05
22:10
flp_svcIncomingBytes{cloud="cloud-0b0b016b-036e-4a55-879d-c52a0f975428", cluster="cluster-6

- Incoming packets for a service


Incoming packets for a service
14:35
14:40
14:45
flp_svcIncomingPackets{cloud="cloud-0b0b016b-036e-4a55-879d-c52a0f975428", cluster="cl

- Gateway Incoming/Outgoing rate (that corresponds to the tenant)


GW Incoming/outgoing rate
60
50
40
30
20
10
22:00
22:05
22:10
incoming rate in bytes 1,tun_gw2
incoming rate in bytes 2,tun_gw1

To view a specific chart in detail, hover over the chart name and click the **arrow**, then click **View**. You can click the **clock** icon on the toolbar to set the time range for viewing the application traffic. Also, you can hover over the charts to view the corresponding details.

By default, the dashboard is refreshed every 10 seconds. You can set the refresh frequency by clicking the **arrow** next to the **refresh** icon and selecting the required frequency.

Dashboard configuration file

To configure the dashboard, use this file. For more information, see Viewing application network traffic.

```
# # config file version
apiVersion: 1

providers:
  - name: 'MCNM-HUB'
#   orgId: 1
#   folder: ''
#   folderUid: ''
    type: file
    # <bool> disable dashboard deletion
    disableDeletion: false
    # <int> how often Grafana will scan for changed dashboards
    updateIntervalSeconds: 10
    # <bool> allow updating provisioned dashboards from the UI
    allowUiUpdates: false
    options:
      path: /var/lib/grafana/dashboards
      # <bool> use folder names from filesystem to create folders in Grafana
      foldersFromFilesStructure: true
```

Data source configuration file

To configure the data source, use this file. For more information, see Viewing application network traffic.

```
# # config file version
apiVersion: 1

# # list of datasources that should be deleted from the database
deleteDatasources:
   - name: Graphite
     orgId: 1

# # list of datasources to insert/update depending
# # on what's available in the database
datasources:
#   # <string, required> name of the datasource. Required
 - name: MetricsDatasourceManager
#   # <string, required> datasource type. Required
   type: prometheus
#   # <string, required> access mode. direct or proxy. Required
   access: proxy
#   # <int> org id. will default to orgId 1 if not specified
#   orgId: 1
#   # <string> url
   url: $MDM_SERVICE_URL
#   # <string> database user, if used
#   user:
#   # <string> database name, if used
#   database:
#   # <bool> enable/disable basic auth
#   basicAuth:
```

```
#   # <string> basic auth username
#   basicAuthUser:
#   # <bool> enable/disable with credentials headers
#   withCredentials:
#   # <bool> mark as default datasource. Max one per org
   isDefault: true
#   # <map> fields that will be converted to json and stored in json_data
   jsonData:
#       graphiteVersion: "1.1"
#       tlsAuth: true
#       tlsAuthWithCACert: true
      httpHeaderName1: "Authorization"
      timeInterval: 30s
#   # <string> json object of data that will be encrypted.
   secureJsonData:
#     tlsCACert: "..."
#     tlsClientCert: "..."
#     tlsClientKey: "..."
#     # <openshift\kubernetes token example>
     httpHeaderValue1: $MDM_API_KEY
#   version: 1
#   # <bool> allow users to edit datasources from the UI.
   editable: false
```

Dashboard definition file

To configure the dashboard use this file as explained in Viewing application network traffic

```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": {
          "type": "datasource",
          "uid": "grafana"
        },
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "target": {
          "limit": 100,
          "matchAny": false,
          "tags": [],
          "type": "dashboard"
        },
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "fiscalYearStartMonth": 0,
  "graphTooltip": 0,
  "id": 1,
  "links": [],
  "liveNow": false,
  "panels": [
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "binBps"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 9,
        "w": 12,
        "x": 0,
        "y": 0
      },
      "id": 8,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "rate(flp_appOutgoingBytes{app_name=~\"${appName}\", dir=\"in\"}
[$__rate_interval])",
          "interval": "",
          "legendFormat": "App Name: {{app_name}}, App IP: {{srcIP}}, Cluster 
Type: {{app_cluster_type}}, Cluster Name: {{app_cluster_name}}, Location: 
{{app_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "Outgoing bytes for an app",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "binBps"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 12,
        "y": 0
      },
      "id": 6,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "rate(flp_svcIncomingBytes{svc_name=~\"${svcName}\", 
dir=\"out\"}[$__rate_interval])",
          "interval": "",
          "legendFormat": "Service Name: {{svc_name}}, Service IP: {{dstIP}} 
Cluster Type: {{svc_cluster_type}}, Cluster Name: {{svc_cluster_name}}, Location: 
{{svc_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "Incoming bytes for a service",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "none"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 12,
        "y": 8
      },
      "id": 4,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "rate(flp_svcIncomingPackets{svc_name=~\"${svcName}\",
dir=\"out\"}[$__rate_interval])",
          "interval": "",
          "legendFormat": "Service Name: {{svc_name}}, Service IP: {{dstIP}}
Cluster Type: {{svc_cluster_type}}, Cluster Name: {{svc_cluster_name}}, Location:
{{svc_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "Incoming packets for a service",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "none"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 9,
        "w": 12,
        "x": 0,
        "y": 9
      },
      "id": 2,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "rate(flp_appOutgoingPackets{app_name=~\"${appName}\", 
dir=\"in\"}[$__rate_interval])",
          "interval": "",
          "legendFormat": "App Name: {{app_name}}, App IP: {{srcIP}}, Cluster 
Type: {{app_cluster_type}}, Cluster Name: {{app_cluster_name}}, Location: 
{{app_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "Outgoing packets for an app",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "percent"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 12,
        "y": 16
      },
      "id": 13,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "pluginVersion": "9.3.6",
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "(gw_memory_MemUsed_bytes/gw_memory_MemTotal_bytes)*100",
          "interval": "",
          "legendFormat": "Gateway Name: {{gw_name}}, IP: {{gw_ip}}, Cluster Type: 
{{gw_cluster_type}}, Cluster Name: {{gw_cluster_name}}, Location: 
{{gw_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "GW Memory Utilization",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "binBps"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 0,
        "y": 18
      },
      "id": 12,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": 
"rate(gw_network_logical_incoming_bytes_total[$__rate_interval])",
          "interval": "",
          "legendFormat": "Incoming rate in bytes - Gateway Name: {{gw_name}}, IP: 
{{gw_ip}}, Tunnel ID: {{tunnel_id}}, Cluster Type: {{gw_cluster_type}}, Cluster 
Name: {{gw_cluster_name}}, Location: {{gw_location_name}}",
          "range": true,
          "refId": "A"
        },
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "rate(gw_network_logical_outgoing_bytes_total[1m])",
          "hide": false,
          "interval": "",
          "legendFormat": "Outgoing rate in bytes - Gateway Name: {{gw_name}}, IP: 
{{gw_ip}}, Tunnel ID: {{tunnel_id}}, Cluster Type: {{gw_cluster_type}}, Cluster 
Name: {{gw_cluster_name}}, Location: {{gw_location_name}}",
          "range": true,
          "refId": "B"
        }
      ],
      "title": "GW Incoming/outgoing rate",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "µs"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 12,
        "y": 24
      },
      "id": 16,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "expr": "gw_rtt_avg_us",
          "interval": "",
          "legendFormat": "Gateway Name: {{gw_name}}, IP: {{gw_ip}}, Peer IP: 
{{rtt_peer_ip}}, Cluster Type: {{gw_cluster_type}}, Cluster Name: 
{{gw_cluster_name}}, Location: {{gw_location_name}} \n Remote Gateway Name: 
{{gw_name}}, Remote IP: {{gw_ip}}, Remote Peer IP: {{rtt_peer_ip}}, Remote Cluster 
Type: {{gw_cluster_type}}, Remote Cluster Name: {{gw_cluster_name}}, Remote 
Location: {{gw_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "GW RTT",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "mode": "palette-classic"
          },
          "custom": {
            "axisCenteredZero": false,
            "axisColorMode": "text",
            "axisLabel": "",
            "axisPlacement": "auto",
            "barAlignment": 0,
            "drawStyle": "line",
            "fillOpacity": 0,
            "gradientMode": "none",
            "hideFrom": {
              "legend": false,
              "tooltip": false,
              "viz": false
            },
            "lineInterpolation": "linear",
            "lineWidth": 1,
            "pointSize": 5,
            "scaleDistribution": {
              "type": "linear"
            },
            "showPoints": "auto",
            "spanNulls": false,
            "stacking": {
              "group": "A",
              "mode": "none"
            },
            "thresholdsStyle": {
              "mode": "off"
            }
          },
          "mappings": [],
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "percent"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 0,
        "y": 26
      },
      "id": 17,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "tooltip": {
          "mode": "single",
          "sort": "none"
        }
      },
      "pluginVersion": "9.3.6",
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "(gw_disk_used_bytes/gw_disk_total_bytes)*100",
          "interval": "",
          "legendFormat": "Gateway Name: {{gw_name}}, IP: {{gw_ip}}, Cluster Type:
{{gw_cluster_type}}, Cluster Name: {{gw_cluster_name}}, Location:
{{gw_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "GW Disk Utilization",
      "type": "timeseries"
    },
    {
      "datasource": "MetricsDatasourceManager",
      "fieldConfig": {
        "defaults": {
          "color": {
            "fixedColor": "#73BF69",
            "mode": "thresholds"
          },
          "mappings": [],
          "max": 100,
          "min": 0,
          "thresholds": {
            "mode": "absolute",
            "steps": [
              {
                "color": "green",
                "value": null
              },
              {
                "color": "orange",
                "value": 50
              },
              {
                "color": "red",
                "value": 80
              }
            ]
          },
          "unit": "percent"
        },
        "overrides": []
      },
      "gridPos": {
        "h": 8,
        "w": 12,
        "x": 0,
        "y": 34
      },
      "id": 14,
      "options": {
        "legend": {
          "calcs": [],
          "displayMode": "list",
          "placement": "bottom",
          "showLegend": true
        },
        "orientation": "auto",
        "reduceOptions": {
          "calcs": [
            "lastNotNull"
          ],
          "fields": "",
          "values": false
        },
        "showThresholdLabels": false,
        "showThresholdMarkers": true,
        "text": {},
        "tooltip": {
          "mode": "single"
        }
      },
      "pluginVersion": "9.5.2",
      "targets": [
        {
          "datasource": "MetricsDatasourceManager",
          "editorMode": "code",
          "exemplar": true,
          "expr": "gw_cpu_util_avg_user",
          "interval": "",
          "legendFormat": "Gateway Name: {{gw_name}}, IP: {{gw_ip}}, Cluster Type:
{{app_cluster_type}}, Cluster Name: {{app_cluster_name}}, Location:
{{gw_location_name}}",
          "range": true,
          "refId": "A"
        }
      ],
      "title": "GW CPU Utilization",
      "type": "gauge"
    }
  ],
  "refresh": "",
  "schemaVersion": 38,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": [
      {
        "current": {
          "isNone": true,
          "selected": false,
          "text": "None",
          "value": ""
        },
        "datasource": "MetricsDatasourceManager",
        "definition": "label_values(app_name)",
        "hide": 0,
        "includeAll": false,
        "label": "App Name",
        "multi": true,
        "name": "appName",
        "options": [],
        "query": {
          "query": "label_values(app_name)",
          "refId": "StandardVariableQuery"
        },
        "refresh": 1,
        "regex": "",
        "skipUrlSync": false,
        "sort": 0,
        "type": "query"
      },
      {
        "current": {
          "isNone": true,
          "selected": false,
          "text": "None",
          "value": ""
        },
        "datasource": "MetricsDatasourceManager",
        "definition": "label_values(svc_name)",
        "hide": 0,
        "includeAll": false,
        "label": "Service Name",
        "multi": true,
        "name": "svcName",
        "options": [],
        "query": {
          "query": "label_values(svc_name)",
          "refId": "StandardVariableQuery"
        },
        "refresh": 1,
        "regex": "",
        "skipUrlSync": false,
        "sort": 0,
        "type": "query"
      }
    ]
  },
  "time": {
    "from": "now-5m",
    "to": "now"
  },
  "timepicker": {},
  "timezone": "",
  "title": "MCNM Dashboard",
  "uid": "",
  "version": "",
  "weekStart": ""
}
```

Using the Mesh console

The Mesh console enables you to:

View topology:

- Visualize application resources and infrastructure resources through the canvas view.
- View the details of the resources such as locations, applications, services, and edge gateways at a glance, through corresponding side panels.
- Understand relationships, connections, and diagnose problems by viewing key details of your network by canvas view in topology.

Manage resources:

- Register and view applications, services, and locations so Mesh can manager their connectivity.
- View deployment environments and choose which Mesh should manage the network connectivity for.
- Create resource groups to set access for all of the resources in a group by a single action.
- View details of created applications, services, deployment environments, locations, and clouds in the list or card view.
- View managed and unmanaged resources.
- Filter applications by labels.
- Filter deployment environments by:
  - Cloud
  - Location
  - Type
  - Labels
- Filter locations by:
  - Cloud
  - Type
  - City
  - Country
  - Labels
- Filter clouds by the provider

Manage policies:

- Create, view, delete connection policies to control the network connectivity access between applications and services.
- Filter the policies by application, service, and labels.

Use the Grafana dashboard to:

- Visualize real-time application network traffic through Grafana.
- View charts of outgoing bytes and packets for an app, and incoming bytes and packets for a service.
- View the gateway incoming or outgoing rate that corresponds to the tenant.

Logging in to the Mesh console

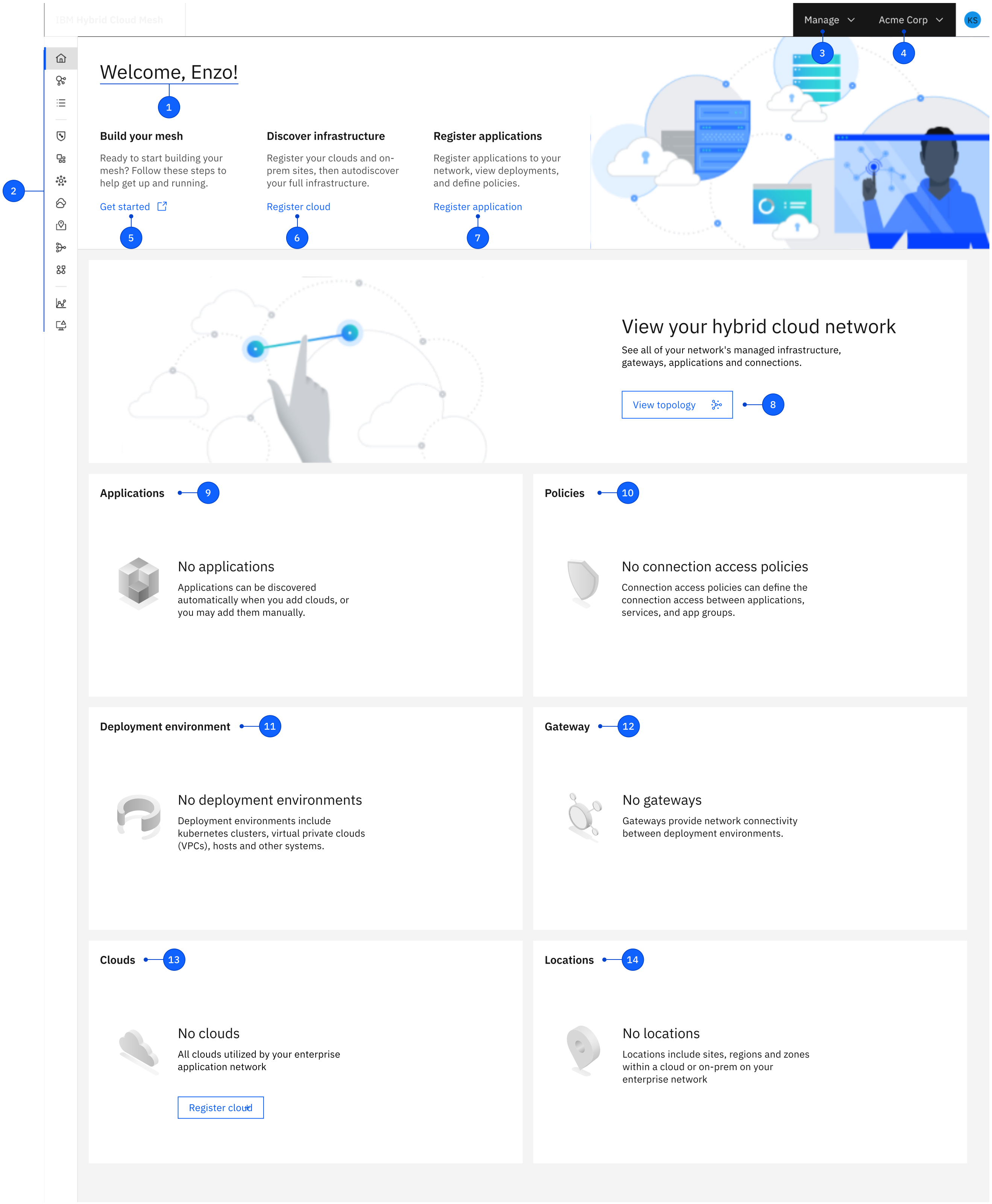
Follow these steps to log in to the Mesh console with Single Sign-On (SSO):

1. Using a supported web browser, browse the IBM® Hybrid Cloud Mesh console.
2. Click **Login** to log in to **Mesh**.
3. Enter your IBMid on the IBM login page and click **Continue**.
4. Sign in with your identity provider login credentials.

The Mesh console Home page

Figure 1 shows the Mesh console Home page.

Figure 1. Mesh console Home page



Welcome, Enzo!

Build your mesh

Ready to start building your mesh? Follow these steps to help get up and running.

Get started

Discover infrastructure

Register your clouds and on-prem sites, then autodiscover your full infrastructure.

Register cloud

Register applications

Register applications to your network, view deployments, and define policies.

Register application

View your hybrid cloud network

See all of your network's managed infrastructure, gateways, applications and connections.

View topology

Applications

No applications

Applications can be discovered automatically when you add clouds, or you may add them manually.

Policies

No connection access policies

Connection access policies can define the connection access between applications, services, and app groups.

Deployment environment

No deployment environments

Deployment environments include kubernetes clusters, virtual private clouds (VPCs), hosts and other systems.

Gateway

No gateways

Gateways provide network connectivity between deployment environments.

Clouds

No clouds

All clouds utilized by your enterprise application network

Register cloud

Locations

No locations

Locations include sites, regions and zones within a cloud or on-prem on your enterprise network

Table 1. Describes the elements on the Mesh console Home page.

Table 1. Mesh console Home page elements

Table 1. Mesh console Home page elements

Label	Element	Description
1	User Logged in	Displays the username of the user that is logged in.

Label	Element	Description
2	Navigation bar	Displays the navigation options available to the logged in user. For more information, see Table 2.
3	Manage drop-down	Displays the secrets, identities, roles and resource groups option. Click ⌄ to view the various options.
4	User drop-down	Displays the user list. Click ⌄ to select the user from the dropdown list.
5	Getting started	Click **Getting started** to view the documentation on getting starting steps.
6	Register cloud	Click **Register cloud** to register a new cloud.
7	Register application	Click **Register application** to register a new application.
8	Network topology	Displays the interconnected pattern of network elements. Click **View topology** to view the Network topology map that displays the locations, deployment environments, associated applications and services, and established connections within your enterprise network.
9	Applications	Displays the number of applications on the user's enterprise network and a list of application names and the last update date. Click **View all**, to view all the applications and services listed on the Applications page. You can also click on the name of an application to view the application details on the Application details page.
10	Policies	Displays the number of policies on the user's enterprise network and list of policy names and the last update date. Click **View all** to view the policies you created, listed on the Connection access policies page. You can also click on the name of a policy to view the policy details on the Policy details page.
11	Deployment environments	Displays the number of Environments on the user's enterprise network and a list of environment names and the last update date. Click **View all** to view all the deployment environments listed on the Environments page. You can also click on the name of a deployment environment to view the deployment environment details on the Deployment environment details page.
12	Gateways	Displays the number of registered gateways. Click **View all** to view all the gateways listed on the gateways page. You can also click on the name of a gateway to view the cloud details on the cloud details page.
13	Clouds	Displays the number of registered clouds. Click **View all** to view all the clouds listed on the Clouds page. Click on **Register cloud** to register a new cloud. You can also click on the name of a cloud to view the cloud details on the cloud details page.
14	Locations	Displays the number of registered locations. Click **View all** to view all the locations listed on the Locations page. You can also click on the name of a location to view the location details on the location details page.

Table 2 describes the navigation options available on the Mesh console.

Table 2 Navigation bar elements

Table 2. Navigation bar elements

Element	Description
Home ⌂	Click ⌂ to navigate to the Home page.

Element	Description
Topology	Click to navigate to the Topology page to view the interconnected pattern of network elements across clouds.
Policies	Click to navigate to the Connection access policies page, listing the policies that define the connection access between applications, services, and deployment groups.
Applications & Services	Click to navigate to the Applications page, listing all the applications and services on the user's enterprise application network.
Environments	Click to navigate to the Deployment environments page, listing the Kubernetes clusters, virtual private clouds (VPCs), hosts and other targets where applications run.
Cloud	Click to navigate to the Clouds page to view the registered clouds.
Locations	Click to navigate to the Locations page to view cloud locations.
Gateways	Click to navigate to the Gateways page to view all the registered edge and waypoint gateways along with the associated deployment environments.
Network segments	Click to navigate to the Network segment page to view all the registered network segments.
Events	Click to navigate to the Events page to view the system generated events.

Using the IBM® Hybrid Cloud Mesh console topology view

Use the Mesh console topology view to visualize the key details and relationships between your locations, deployment environments, gateways, and associated applications and services.

Follow these steps to explore the topology view:

1. Double-click to view the associated deployment environment. Different types of deployment environments are represented by different icons, for example, represents cluster type, represents VPC type, represents node type and represents infra only VPC type. Single-click on each of these icons to display the corresponding side panels displaying real-time details.
2. Double-click the associated deployment environment icon to view the associated gateways displaying the connection to the other locations and to view the associated partitions. Double-click on the partitions to display the associated applications.Thick lines between gateways represent a connection between gateways. Single-click on the gateway icon to display the gateway side panel displaying real-time gateway details.
3. Double-click to view the associated port. Straight lines between application and service represent data flow. Single-click on the dataflow line between an application and service to display the application to service connection side panel displaying real-time data insights. Dashed lines display relations between entities.
4. Click and check the **Policy connections** checkbox to display the existing policies represented by the dotted lines between application and services. If there is dataflow between application and service the dotted line is overlapped by the straight data flow line.

Click and select from the displayed options to view the associated locations, deployment environments, partitions, applications and services. You can hover over the resources to view the tooltip information.

Expand the **Managed** dropdown to view managed or unmanaged topology elements by clicking the corresponding option. Expand the **Network Segment** dropdown and select the corresponding network segment checkbox to filter the topology elements associated with the selected network segment.

You can also rearrange the topology elements by dragging the elements. Double-click twice on any element in the topology to collapse it. You can pinch zoom in or out from your laptop touchpad, and you can also use the mouse scroll wheel to zoom in and out if using a mouse.

Note: Map view is deprecated until further notice. It will be considered for reintroduction in the future.

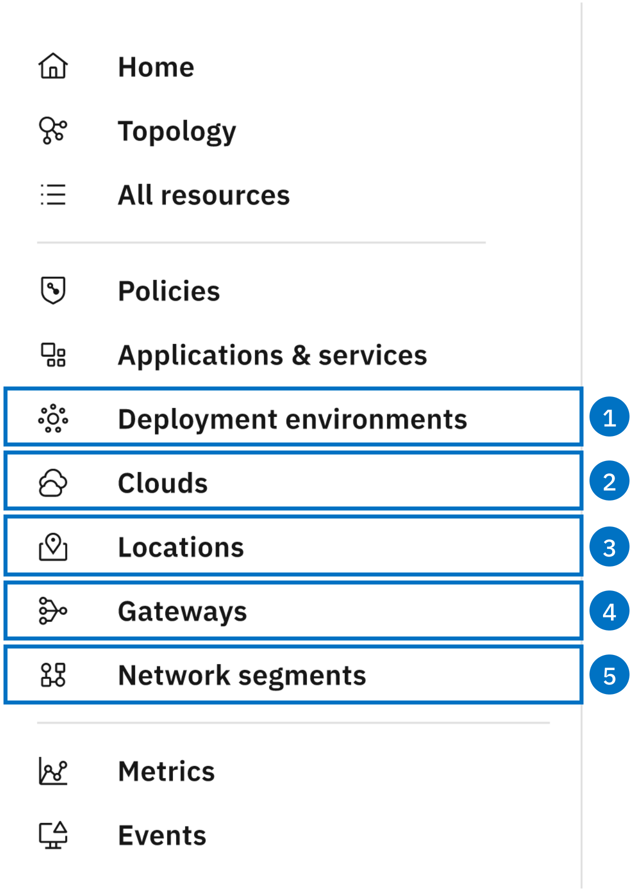
Working with infrastructure resources

You initially make IBM Hybrid Cloud Mesh (Mesh) aware of your cloud infrastructure by asking Mesh to discover it or by manually registering it.

Then, you can use the Mesh to view your infrastructure resources:

Figure 1 shows the Navigation panel.

Figure 1. Mesh console panel



- Click Clouds ☁ (labelled 2) on the Navigation pane to navigate to the Clouds page. Use the Mesh console clouds list to view the clouds you have manually registered or auto-deployed with Mesh. Click on a cloud name to further view cloud details along with associated locations and deployment environment, under each location on cloud details page. For a cloud with auto-discovery off, you can add deployment environment under a location from cloud details page. You can toggle cloud resources auto-discovery on to make IBM Hybrid Cloud Mesh (Mesh) aware of your cloud infrastructure.

- Click Locations (labelled 3) on the Navigation pane to navigate to the Locations page. Use the Mesh console cloud locations list to view the locations you have registered with Mesh. From here you can drill down to see the registered deployment environments within a location.
- Click Deployment environments (labelled 1) on the Navigation pane to navigate to the Deployment environments page. Use the Mesh console deployment environments page to view all of the deployment environments you have registered with Mesh. These are the clusters, VPCs, or VMs where your applications and services are running. All namespaces managed by a Mesh gateway belong to the default network segment. See working with Red Hat® Service Interconnect for info on how to create network segments for use with the Red Hat® Service Interconnect tech preview.
- Click Gateways (labelled 4) on the Navigation pane to navigate to the Gateways page. Use the Mesh console gateways list page to see all the registered Mesh edge gateways, Red Hat® Service Interconnect, Mesh waypoint gateways along with the associated deployment environments.
- Click Network segment (labelled 5) on the Navigation pane to navigate to the Network segment page. Use the Mesh console gateways list page to see all the network segments registered. Currently, Mesh edge and Mesh waypoint gateways carry a default network segment and no new network segments can be created for them. New network segments can only be created for Red Hat® Service Interconnect gateways. See working with Red Hat® Service Interconnect for information on the Red Hat® Service Interconnect tech preview feature.

Manage locations, deployment environments, and partitions as follows:

- To unmanage a location, unmanage or remove all the deployment environments under that location.
- To unmanage a deployment environment, turn off the auto-discover and unmanage the namespaces.
- To unmanage a partition (namespace/ security groups), turn off auto-discover and delete all application deployments. To delete all application deployments, delete all instances and service endpoints.

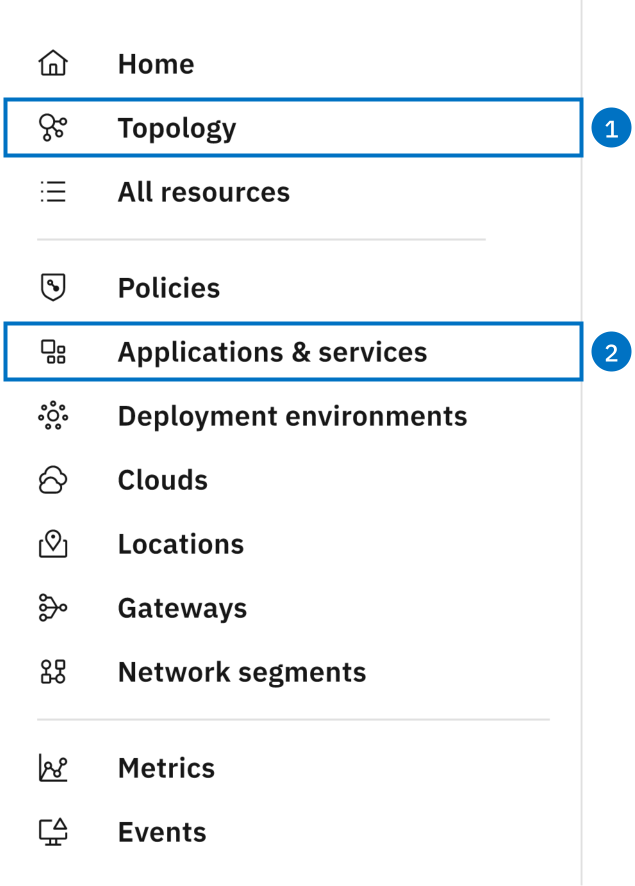
Working with application resources

You initially make IBM Hybrid Cloud Mesh (Mesh) aware of your applications and services by discovering cluster namespaces and application or by manually registering them.

Then, you can use the IBM® Hybrid Cloud Mesh console to view your applications and services.

Figure 1 shows the Navigation panel.

Figure 1. Navigation pane



- Click Topology (labelled 1) on the Navigation pane to navigate to the Topology view. Use the Mesh console topology view to see all the relationships between your applications and services and the corresponding details.
- Click Applications & Services (labelled 2) on the Navigation pane to navigate to the Applications list page. Use the Mesh console applications list page to view all of your applications in Table view or Card view. Filter the applications by associated labels. Click on an application to see the services it provides and the deployment environments the services are running in. All applications must belong to a network segment. The name of the application must be unique. All deployments of the application must be within deployment environments or partitions in the same network segment as the application.

Working with the API

This document provides an overview of the IBM Hybrid Cloud Mesh (Mesh) APIs and how to use them. Examples in **`curl`** and Python are shown.

Prerequisites

- Create an API key: Managing API keys
- Set the API key in a variable for subsequent steps:

```
MESH_API_KEY=<your-api-key>
```

- Set the Mesh API base URL in a variable for subsequent steps (replace `<console-url>` with https://app.hybridcloudmesh.ibm.com):

```
MESH_API_URL=<console-url>/api/v1
```

- Review the Mesh API spec

Using the API with `curl`

The API output is in JSON format. It is easier to view it if you install the jq command.

Get the cloud resources:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/nets/cloud | jq
```

Get the cluster resources in the cloud named **`IBM-Cloud`**:

```
cloud_id="$(curl -sSLf -H "Authorization: $MESH_API_KEY" $MESH_API_URL/nets/cloud
| jq -r '.clouds[] | select(.name == "IBM-Cloud") | .resource_id')"
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/nets/cloud/$cloud_id/cluster | jq
```

Register a cloud resource by sending the cloud fields to the **`stdin`** of **`curl`**:

```
cat << EOM | curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY" -X POST
-H Content-Type:application/json -d @- $MESH_API_URL/nets/cloud | jq
{
  "name": "mycloud",
  "type": "other",
  "is_private": true
}
EOM
```

Get the application resources:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/apps/application | jq
```

Register an application resource by sending the application fields to the **`stdin`** of **`curl`**:

```
cat << EOM | curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY" -X POST
-H Content-Type:application/json -d @- $MESH_API_URL/apps/application | jq
{
  "name": "another-app",
  "app_identity": "another-app.my.domain.palmetto.ibm.com",
  "labels": [
    "app:exampleWeb",
    "cloud:aws"
    ]
}
EOM
```

Get the policy resources:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/apps/policy | jq
```

Get the identity (user) resources:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/auth/identity | jq
```

Get the event resources:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
$MESH_API_URL/events/event | jq
```

Get the resource topology:

```
curl -sSLf -w %{http_code} -H "Authorization: $MESH_API_KEY"
"$MESH_API_URL/topology/topology?visibility=all" | jq
```

Using the API with Python

To use the Mesh API from python, use the requests package. Install it with pip:

```
pip install requests
```

Use the requests package to make calls to an API endpoint. The following example shows a simple **get applications** call in python (replace **<console-url>** with https://app.hybridcloudmesh.ibm.com):

```
import requests
MESH_API_URL = "<console-url>/api/v1"
MESH_API_KEY = "<your-api-key>"

# get applications
r = requests.get(MESH_API_URL + "apps/application", headers={'Authorization':
MESH_API_KEY})

print(r.status_code)
print(r.json())
```

Administration guide

In IBM Hybrid Cloud Mesh (Mesh), a tenant is created for the customer and a user is designated as a tenant admin. A user with the tenant administrator role can create new users and assign roles and access permissions within their tenant.

To manage user access, perform the following:

- Adding users
- Managing user roles, permissions, and resource groups
- Viewing events

Adding users

To add users and give them appropriate privileges, the users must create an IBMid and subscribe to IBM Hybrid Cloud Mesh (Mesh) at My IBM.

Add a user (identity)

To add a user, complete the following steps on the Mesh console home page:

1. Click **Manage** and then click **Identities** in the dropdown list.
2. Click **Create identity** to add a user to the system and define its role.
    1. Enter the name and email ID for the new identity, and click **Next**.
    2. Select the roles that you want to assign to the new identity, and click **Add +**. You can assign one or more roles to the identity.
3. Click **Create**.

Create a role with permissions

To create a role with permissions, complete the following steps on the Mesh console home page:

1. Click **Manage** and then click **Roles** in the dropdown list.

2. Click **Create role** and specify the details of the role you want to create.

    1. Enter the name and description for the new role, and click **Next**.
    2. Define the permissions for the role by selecting the required permissions from the Resource type and the Resource group.

    When setting permissions by resource group, you have the option to grant permissions at either the group level or the resource level for each resource group. If you grant permissions at the resource level, the read permission for that group is automatically granted.

3. Click **Create**.

You can assign a role to one or more users. When you assign a role to a user, it gives the user all the permissions that are contained in that role.

Two types of Resource Groups are there, Application Groups and Infrastructure Groups. Application Groups contain the resource types such as Applications, Policies, and Deployment Sets. Infrastructure Groups contain the Resource Types such as Gateways, Deployment Environments, Partitions, and Clouds. For more information, see Managing user roles, permissions, and resource groups.

Managing user roles, permissions, resource groups, and secrets

IBM Hybrid Cloud Mesh (Mesh) provides Resource Based Access Control (RBAC) of system resource definitions such as clouds, identities, applications, and gateways. An RBAC system enforces authorization policies and manages abstract role definitions that describe the set of permissions that a role grants.

Creating roles with permissions

when you create a role, you can add multiple permissions to the role. For example, to grant a user permission to manage the authorization-related resources, create a role called `RBAC-manager`.

- You can assign one or more roles to a user. Similarly, you can assign a role to one or more users.
- A user with only assigned roles has permissions in the system.
- A user with administrator role can add users, create roles, and manage user access.

When you assign a role to a user, it gives the user all the permissions that are contained in that role.

For more information about how to add users and create roles with permissions, see Adding users.

Default roles

The following are the default roles for which you can't modify the permissions:

- **Admin**: A user with the Admin role has permissions to Create, Read, Update, Delete, and Admin all the resources in the system. The Admin role is typically used in POCs where you don't need Access Control.
- **RBACManager**: A user with the RBACManager role has permissions to Create, Read, Update, Delete, and Admin the Identities and Roles in the system. You can't create a new role with these specific permissions.
- **CloudOps**: A user with the CloudOps role has permissions to Create, Read, Update, Delete, and Admin the `Default_Infrastructure_Group`, and permission to Read the `Default_Application_Group`.
- **DevOps**: A user with the DevOps role has permissions to Create, Read, Update, Delete, and Admin the `Default_Application_Group`, and permission to Read the `Default_Infrastructure_Group`.

The CloudOps and DevOps roles are typically used if you are using only the default resource groups.

Creating resource groups

Resource groups enable administrators to create group of resources and grant permission to all the resources in the group. Resource groups provide a more scalable approach than managing hundreds or thousands of individual permissions. If a set of resources is placed in a resource group, the administrator can give access to users for all those resources in a single command.

The following are the types of resource groups:

- **Infrastructure resource group**: A related group of deployment environments, partitions, gateways, and other infrastructure resources.
- **Applications resource group**: A related group of applications, services, deployments, instances, service-endpoints, and policies.

The two default resource groups that are called the `Default_Application_Group` has the resource ID `default-app`, and the `Default_Infrastructure_Group` has the resource ID `default-infra`. When you create an individual resource without specifying the resource group, it is added to one of the default resource groups. The default resource group to which it is added depends on whether the resource is an infrastructure-related resource or an application-related resource.

To create a resource group, complete the following steps on the Mesh console home page:

1. Click **Manage** and then click **Resource Groups** in the dropdown list.
2. Click **Create resource group**. Enter the name of the resource group, and choose the resource group type that you want to create.
3. Click **Create**.

To create resource groups from the CLI, see Creating resource Groups from the CLI.

Creating roles with resource groups

After you group your resources into resource groups, you can create roles that have specific types of access to all the resources in those groups.

Registering a secret

Before you register a secret, you must configure the secrets manager, and set the secrets manager type to **`external`** or **`internal`**. By default, the secrets manager type is set to **`internal`**. For more information about how to configure the secrets manager, see Managing secrets.

To register a secret, complete the following steps on the Mesh console home page:

1. Click **Manage** and then click **Secrets** in the dropdown list.
2. Click **Register secrets** and specify the details of the secret that you want to create.
    1. Enter the name of the secret, and select the secret type.
    2. Enter the API key or Access key details.
3. Click **Register**.

Creating resource Groups from the CLI

You can create resource groups that contain resources that a specific group can access.

For example, run a command like this to create an Application resource group that is called Group_A that contains the resources that are accessible to Group_A:

```
cat << EOM | palmctl create resource-group -f-
name: Group_A
description: app resources owned by Group_A
type: application
EOM
```

If you want to create an Infrastructure resource group, set the **`type`** field to **`infrastructure`**.

You can add a resource to only one resource group.

You can see all your resource groups by running the following command:

```
palmctl get resource-groups
```

When you create resources, you can add them to the resource group that you already created.

For example, run a command like this to create an application and add it to the resource group Group_A:

```
cat << EOM | palmctl create application -f -
name: my-app
description: A sample application
app_identity: my-app.palmetto.ibm.com
resource_group_id: <resource_id of the resource group>
EOM
```

`<resource_id of the resource group>` is the **`resource_id`** of Group_A.

The users with access to Group_A automatically have access to the application.

Querying resources

When querying for a specific resource, you might receive one of the following responses:

- The current state of that resource.
- A `no permission` error, which indicates that you don't have sufficient permission to access the resource.

When querying for a list of resources, you might receive one of the following responses:

- An empty list, if no resources that are matching the query.
- A list of resources that match the query and to which you have access.
- A `no permission` error, which indicates that you don't have sufficient permission to access any of the matching resources.

Viewing events

Viewing events using the IBM® Hybrid Cloud Mesh console

Click on the IBM® Hybrid Cloud Mesh console navigation pane to view events related to your enterprise application network.

You can view events severity levels, status and other events properties such as event message, resource name, assigned to and time of event, to get more insights.

Click on the checkbox next to an event on the events list to delete or close the event.

Events can have statuses as open, asssigned, or closed.

Events with open status are displayed in bold on the events list. Click on the ⌄ to view the event details.

Click the event message displayed on the events list to view the event details page.

If an event is in open status, you can assign it to a user or close it from the events details page. If an event is assigned, you can reassign it to another user or close it. If an event is closed, you can reopen it.

Viewing events using the CLI

Events are generated by IBM Hybrid Cloud Mesh for things that happen in the system that users need to be aware of.

Anatomy of an event

An event generated by Mesh has these properties describing the event:

Table 1. Event properties

Properties	Description
`message`	Message giving information about the event.
`severity`	Impact on the system and the potential errors it can cause. One of: `critical`, `major`, `minor`, `information`, `cleared`.
`category`	System category of the event. One of: `system`, `recommendation`.
`created_at`	Date when it was created.
`correlated`	A cleared event uses this field to point to the event it is clearing.

Properties	Description
`resource_type` and `resource_instance`	The resource this event is about (app, cloud, ...)

The severity levels have these definitions:

Table 2. Event security levels

Severity	Definition	Examples
`critical`	An indication of a severe service problem, affecting core functionality, that requires immediate attention and resolution, regardless of the time of day.	• A gateway is down • Audit log forwarding to the customer's S3 storage is failing
`major`	An indication of a problem that is having (or could have) significant impact on service, and steps must be taken as soon as possible because the affected service has degraded drastically and is in danger of being lost completely.	• A gateway's traffic forwarding performance is significantly lower than expected • A gateway is almost out of memory
`minor`	An indication of a problem that does not have a serious impact on service. Instead, it affects only convenience for the users, there is a way to work around the problem, or the system is operating sub-optimally. This does not require immediate attention.	A cloud location was discovered for which there are no GPS coordinates
`information`	An event that is a result of normal operation of the system.	• A new gateway image is available for deployment • Infrastructure discovery has completed • A gateway deployment has completed
`cleared`	Indicates that the previous event referenced in the `correlated` field has automatically resolved.	A gateway that was unreachable is now reachable

Events have additional properties to set when managing them:

Table 3. Additional event properties

Additional Properties	Description
`assigned_user`	The user assigned to handle this event (email of the assigned user).
`comment`	Free-form comments related to handling this event. Additional description or comments set by user.

Additional Properties	Description
`labels`	User-defined labels. For example: categorize the event or set its state.
`is_handled`	Set to true when the event has been addressed.

Viewing events

Use the following command to view all of the events that have not been handled yet:

```
palmctl get events --is-handled false
```

Further filter the event list by using one or more of the following flags:

- View a specific event: `--id <event-id>`
    - Where `<event-id>` is the `resource_id` property of the event
- View events created before this date and time: `--date <RFC3339-date>`
    - Example date and time: `'2023-01-04T00:00:00Z'`
- View events of a specific category: `--category <category-name>`
- View events with this severity or higher: `--severity <severity-name>`
- View events with this severity or lower: `--highest-severity <severity-name>`
- View events assigned to this user: `--user <user-name>`
- View events that have been handled or not: `--is-handled {true|false}`
- View events related to this resource: `--resource-instance <resource-instance>`
    - Where `<resource-instance>` is `resource-id` of the other resource this event is about

Managing events

While handling an event, set any of the modifiable properties described previously using the `patch` sub-command. For example, to assign an event to Joe:

```
echo 'assigned_user: Joe' | palmctl patch event -f -
```

Deleting events

After an event has been addressed, keep the event for historical purposes and set it as handled:

```
echo 'is_handled: true' | palmctl patch event <event-id> -f -
```

Or, simply delete the event:

```
palmctl delete event <event-id>
```

Validating Red Hat Service Interconnect container signatures

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Mesh uses the Open Horizon agent to download and install Red Hat Service Interconnect images in your application clusters. Open Horizon automatically checks the signature of each container. If a signature mismatch is detected, the Red Hat Service Interconnect gateway is not deployed.

If you want to manually verify that your Red Hat Service Interconnect images are authentic, use this procedure to check the container images.

Before you begin

Run the **install_cosign.sh** script to install Cosign. For more information, see Installation.

Procedure

Use the IBM Cloud® Container Registry (ICR) to verify the signatures of the Red Hat Service Interconnect images that are running as containers in your cloud-native infrastructure. Complete the following steps:

1. Get the Secure Hash Algorithm (SHA) of the images in Kubernetes infrastructure by running the following command:

   ```
   > kubectl describe <pod-name> | grep 'Image'
   ```

   Output similar to the following example is shown:

   ```
   Image:          docker-na-private.artifactory.swg-devops.com/ccs-mcnm-team-
   testing-docker-local/gw-control-skupper:0.0.0-def456
   Image ID:       docker-na-private.artifactory.swg-devops.com/ccs-mcnm-team-
   production-docker-local/gw-control-skupper@sha256:abc123
   ```

   abc123 is the SHA in this example output.

2. Pull the signed images from the ICR and inspect the SHA to match them with the SHA that you obtained from your infrastructure in the previous step. Run a command like this:

   ```
   > docker pull icr.io/imm-production/gw-control-skupper:0.0.0-def456
   ...
   > docker inspect --format "{{.Id}}" icr.io/imm-production/gw-control-
   skupper:0.0.0-def456
   sha256:abc123
   ```

3. Copy the following public key into the **cosign.pub** file, which is used for signing images.

   ```
   -----BEGIN PUBLIC KEY-----
   MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEtTKXKRfuX2aDUG6e6oB1mv9bMp/h\n2amIbRSJFcQ
   dzNIg0SlIMlQ49dpCWouZY1UfnJTdz0vcZvLUroThmzc/EA==
   -----END PUBLIC KEY-----
   ```

4. Run a command like this to verify the signatures of the images that you pulled from the ICR.

   ```
   cosign verify --key ~/cosign.pub icr.io/imm-production/gw-control-
   skupper:0.0.0-def456 | jq -r
   ```

Working with Mesh gateways

IBM Hybrid Cloud Mesh (Mesh) gateways provide the connectivity between your applications and services.

- Understanding IBM Hybrid Cloud Mesh gateways provides an in-depth look into how gateways connect applications.

  - Gateway network requirements lists the network requirements for Mesh gateway VMs.
- Auto-deploying gateways describes how to auto-deploy gateways in AWS and IBM Cloud® using the Mesh console.
- Deploying on-prem gateways in VMWare OCP describes how to deploy on-prem gateways in OCP running in VMWare vSphere.
- Monitoring gateways describes how to monitor the operational status of gateways.
- Updating gateways describes how to update a gateway when a new image is available.
- Removing gateways describes how to remove gateways when they are no longer needed.
- Deploying gateways in AWS describes how to deploy gateways in AWS using the CLI.
- Deploying gateways in IBM Cloud describes how to deploy gateways in IBM Cloud using the CLI.
- Adding a Cloud Object Storage bucket URL describes how to add a Cloud Object Storage bucket URL in IBM Cloud.

Understanding IBM Hybrid Cloud Mesh (Mesh) gateways

How your applications and services connect using Mesh

Mesh connects application and service endpoints between different *locations*. A location can contain one or more *deployment environments (DE)*. A DE is defined as a set of compute resources in a particular location, which are able to deploy and run applications and services. See Figure 1 in the Overview.

In the case of Kubernetes (K8s) and other environments, a DE can be further sliced into *partitions*. For K8s DEs, partitions map to namespaces. Partitions can, in turn, constitute DEs on their own, and are managed by other devops users.

The Mesh product is a SaaS platform. CloudOps engineers onboard new locations and DEs. From this point, Mesh will deploy and setup one or more gateways (GW) to enable connectivity. Mesh will deploy and configure GWs automatically, when possible.

Devops users, either through the Mesh console, **`palmctl`** command line tool, or REST API, can then step in and start provisioning connectivity for their applications.

The fundamental abstraction to provide connectivity between applications is a set of *connection policies*. A connection policy defines which applications can consume each service, effectively enabling bidirectional connectivity between all the instances of applications and services referenced in that policy. Its logical structure is:

```
FROM < Application | Partition > TO < Service | Partition >
```

Mesh overlay network overview

Mesh creates a layer3 overlay between the GWs within a Mesh tenant. GWs use standard routing protocols and advanced L3/L4 capabilities to provide L3 secured, unproxied, and resilient connectivity to the applications.

The overlay, formed by the GWs, is fully and automatically managed by the Mesh.

Mesh overlay

The Mesh overlay is created by establishing IP tunnels (Geneve, GRE, IPSEC, and so on) between the GWs.

Mesh fully manages the overlay topology and the routing configuration, which can be dynamically adjusted based on the conditions.

On top of these tunnels, routing protocols and BFD, among others, are used to ensure traffic can be forwarded resiliently.

The Mesh core is also designed to allow Traffic Engineering (TE) capabilities for the edge GWs to route through it using Segment Routing (SR) and other advanced techniques.

Mesh edge gateway

The Mesh edge gateway is an advanced L3/L4 router. It is deployed close to a DE, so that traffic can be routed for the applications and services in that DE through the Mesh overlay to applications and services in other DEs.

The GW performs the following functions:

- Policy enforcement: allows only traffic defined by connection policies
- Network Address Translation (1:1): stateless and transparent NAT
- L3 routing, tunneling, and forwarding
- Traffic engineering

Mesh waypoint

Waypoints are L3 routers that have traffic engineering (TE) capabilities and are deployed at strategic locations, without collocated DE, applications, or services. They allow Mesh to route traffic through specific underlay networks, to optimize the traffic flow (for example, latency, stay within certain geographical boundaries, or minimize cost). The utilization of waypoints can be controlled by adjusting the IGP cost set for each gateway connection.

Support for overlapping IP addresses

Mesh provides connectivity between endpoints in private networks. One of the common problems when connecting multiple locations is that they may use overlapping or clashing IP (private) addressing.

Mesh exchanges packets using its own virtual IP address space, unknown to the applications. To achieve this, Mesh assigns unique IP addresses to applications and services to identify them, and uses transparent stateless 1:1 Network Address Translation (NAT), ensuring applications can communicate without problem across locations.

Policy enforcement

Policy enforcements ensure that only traffic allowed by a Mesh connection policy flows through the Mesh overlay.

For security reasons, policy enforcement happens both when traffic ingresses the Mesh overlay and when it egresses, making sure Zero Trust principles are applied.

DNS

After the DE has been onboarded into Mesh and connection policies defined, applications can start consuming a service in the overlay by using a Fully Qualified Domain Name (FQDN) under palmetto.ibm.com.

What's next?

Deploy gateways in the deployment environments where your applications are running. See Auto-deploying Mesh edge gateways.

Gateway network requirements for IBM Hybrid Cloud Mesh

Introduction

The Mesh edge gateway (GW) is packaged as a virtual machine (VM) that can run on different environments, from public cloud providers as AWS or IBM Cloud®, to private on-prem infrastructure on top of VMware vSphere. GW requirements from the different environments are summarized below.

Assumptions

In this document, the generic term workload refers to any source or destination of traffic that is managed by the Mesh GWs. In the specific case of Kubernetes clusters, most of the required changes apply to the pod or the service Classless Inter-Domain Routing (CIDRs), as they are the actual workloads. Other changes apply to the worker nodes where these workloads run, for example, security group rules.

It is assumed that egress traffic is always allowed and that ingress replies on the same connection are also allowed. This is typically the default case on cloud providers. In this context, egress does not only mean traffic to the internet, but also to a private address on the environment where the GW runs.

For cloud providers, it is assumed that this firewall and traffic control is done through security groups and that network access-control-lists (or any equivalent construct) allow all traffic. This is the default configuration on most cloud providers.

This document often uses the terminology used by the cloud providers (for example, security group instead of firewall).

Because on-prem environments can be highly dependent on the corporate infrastructure, this document does not go into detail for specific infrastructure. If the on-prem environment is a Kubernetes cluster that is using the Calico Container Network Interface (CNI), the expected changes are likely to be similar to IKS or ROKS environments.

Infrastructure requirements

Most of the requirements in this section apply to any deployment environment. References to specific environments are indicated by an asterisk (*).

IP reachability

IPs can be reached in multiple ways that typically depend on the deployment environment. The following options are currently supported cloud providers:

- AWS:
  - Deploy the GW on a public network and assign a public IP address to it.
  - Allocate an elastic IP address to the GW
- IBM Cloud:

- Assign a floating IP address to the GW

Firewalls and Security Groups

Gateways and workloads can receive incoming traffic from some known sources or to some pre-defined ports.

GW security group

GWs receive incoming traffic from different sources (for example, workloads, worker nodes, other GWs, and so on). The following table indicates all the expected sources of traffic for any GW.

Table 1.GW security group

Proto	Port	Source	Purpose	Comment
ALL	ALL	Workloads	Traffic forwarding	Includes pod or service CIDRs on K8s environments
TCP	53	0.0.0.0/0	DNS zone transfers	
*TCP	179	Border Gateway Protocol (BGP) Peers	BGP peering	IKS and ROKS only, peer is one (or more) worker nodes
TCP	7777	0.0.0.0/0	SSH	GW's internal firewall restricts access to selected hosts
UDP	53	Workloads	DNS resolution	Includes pod or service CIDRs on K8s environments
UDP	4500	0.0.0.0/0	IPsec and IKEv2	Traffic coming from other GWs on secure overlay
UDP	6081	0.0.0.0/0	GENEVE tunnels	Traffic coming from other GWs on regular overlay

For security reasons, it is advisable to restrict SSH access to bastion hosts or specific IP addresses only, instead of opening that port widely.

Note: Apart from the security group, there is a firewall inside the GW VM that might also restrict SSH access to a selected group of IP addresses.

Configuring security groups for workloads

1. For all deployment environments, allow incoming traffic from the Mesh global range CIDR.

2. Depending on your deployment environment, you must also configure the following port settings:

   - Red Hat OpenShift on IBM Cloud (ROKS) and IBM Cloud Kubernetes Service (IKS)
     Configure TCP Port 179 to allow incoming traffic from Mesh edge gateways so that BGP sessions can be established between the gateways and the worker nodes.
   - Amazon Elastic Kubernetes Service (EKS)
     If the API endpoint is private, or public and private, configure TCP port 443 to allow traffic from Mesh edge gateways so that the gateways can connect to the Kubernetes API for discovery or cluster configuration.

The following table shows the required configuration for the different workloads:

Table 2.Security group settings for workloads

Proto	Port	Source	Purpose	Comment

Proto	Port	Source	Purpose	Comment
ALL	ALL	Mesh CIDR	Requests and responses	Incoming traffic from other Mesh workloads
*TCP	179	Mesh edge gateway private IP	BGP peering	Traffic from IKS and ROKS only
*TCP	443	Mesh edge gateway private IP	EKS API endpoint	Traffic from EKS only. Required if API endpoint is private, or public and private

Routing tables

GW routing table

GWs might need to send traffic to Pod or Service CIDRs that are not directly reachable from outside the cluster. In these cases, it is acceptable to just forward the traffic to one of the worker nodes. It is important to ensure that traffic follows a symmetric path on both the request and response trips, this is why a BGP session is established on IKS or ROKS environments.

Table 3.GW routing table

Destination	Next hop	Purpose	Comment
*Pod CIDR	Worker node	Forward reply back	IKS or ROKS only. BGP used so egress or ingress node is the same
*Service CIDR	Worker node	Forward requests	IKS or ROKS only. Can be different than the one above

Workloads routing tables

Outgoing traffic from a workload (either on the request or response path) has to be sent to the GWs. In some cases, such as IBM Cloud, routes only apply to one zone, so these routes need to be added to all zones where workloads are present.

Table 4.Workloads routing tables

Proto	Destination	Next hop	Purpose	Comment
*ALL	Mesh CIDR	GW priv IP	Request or response from workloads	Traffic to the Mesh overlay shall go through GWs

DNS changes

IBM® Hybrid Cloud Mesh assigns subdomains of the `palmetto.ibm.com` domain to services being exposed on the overlay. These subdomains resolve to the IP addresses on the IBM Hybrid Cloud Mesh global range CIDR. The GW provides Domain Name System (DNS) resolution to workloads only for those services that have a policy allowing incoming traffic to them.

Name resolution set up depends on the environment. When the environment is a Kubernetes cluster, the GW can manage that by itself (but EKS is an exception).

On AWS/EKS, Route53 needs to be configured so that resolution of the `palmetto.ibm.com` goes through the GW for that VPC.

GW-managed changes to the infrastructure

These changes should be applied by the GW itself, so there is no need to perform them explicitly. They apply to the Kubernetes clusters, and are always performed using the Kubernetes server API.

DNS changes

The GW is configured as the DNS server for the `palmetto.ibm.com` domain. Typically, this is done using ConfigMaps on the Kubernetes configuration. The specific changes may depend on the cluster type, such as vanilla Kubernetes versus OpenShift®.

Calico-based environments

Environments that use the Calico CNI, like IKS or ROKS, require several Custom Resource Definitions (CRDs) defined, so that the overall solution works.

IP Pool

Mesh CIDR must be added to a disabled IP pool. Even if the Mesh CIDR is disabled, traffic to this CIDR should be routed through Calico's overlay. To do this, set the `ipipMode` to CrossSubnet.

Note: There must not be any other `IPPool` CRD for any CIDR that overlaps with Mesh CIDR, because this can cause traffic disruption on the Mesh overlay.

BGP Peer

To make egress path requests the same as the ingress path when replies are returned, some specific routes must be installed from the GW into the worker nodes. The worker nodes already use BGP to distribute routing information among them, and it is possible to add the GW as a BGP peer. The easiest way to do this is to establish an eBGP session (thus, the Autonomous System (AS) of the GW has to be different than the one on the cluster). The BGP peer CRD can be used to indicate the remote AS, the remote peer (in this case, the GW's private IP), as well as the worker nodes that will attempt to establish this session.

Autodeploying gateways using the Mesh console

Use the Mesh console to auto-deploy gateways in AWS and IBM Cloud®.

Prerequisites

The user must be assigned a role that has the authority to auto-deploy gateways. To view the roles and their permissions, complete the following steps:

1. Click **Manage** on the Mesh console home page and click **Roles** in the dropdown list.
2. Click a role, for example, **Admin**.
3. Ensure that the role has the following permissions:

- In the Permissions section, on the **By resource type** tab > **Infrastructure** group:
    - `Read` permission for all clouds, all locations, all clusters, all VPCs, and all gateways
    - `Update` permission for all clusters

    - `Create` permission for all gateways
- In the Permissions section, on the **By resource group** tab > **Infrastructure groups** > **Default_Infrastructure_Group**:
    - 'Admin' permission for Group
    - 'Read' permission for Group and Resources

If you are deploying a gateway in IBM Cloud, you must add the URL in your IBM Cloud account for your Cloud Object Storage bucket. The serviceID and the API key that you generated for the Mesh secrets must have permissions to access your service instance for the Cloud Object Storage bucket. For more information, see Adding Cloud Object Storage bucket URL.

Autodeploying a gateway from Cloud details page

To autodeploy a gateway through Mesh console follow these steps:

1. On the **Clouds** page, click a registered cloud from the cloud list.
2. On the **Cloud details** page, select a location and choose an already discovered deployment environment of type cluster inside a VPC. The autodiscover for the cloud should be on and it should be off for cluster inside the VPC. Cluster should be in unmanaged state.
3. Toggle **Managed** to on for the registered cluster.
4. On the Deployment environment configuration tab, click **Connect a gateway**.
5. On the **Connect edge gateway** tab, two options are shown: Existing gateway and Autodeploy. Select **Autodeploy** to autodeploy the new gateway or select **Existing gateway** to connect to the existing deployed gateway from the displayed list.
6. Enter the Gateway name and choose a Compute profile from the **Compute profile** dropdown list.
7. Check the checkbox stating “By selecting the checkbox, you agree to IBM® provisioning a VM to your cloud account which will incur a recurring cost as long as gateway is deployed.”
8. Click **Deploy**. The gateway is connected to the selected deployment environment and the gateway request notification is displayed.
9. Click **Register**. The deployment environment configuration success notification is displayed.

Autodeploying a gateway from Gateway list page

Follow these steps to autodeploy a gateway from Gateway list page:

1. Go to **Gateways** page and click a **Register gateway**.
2. Select a gateway type to autodeploy. You can choose either Mesh edge gateway or Mesh waypoint gateway. Currently, autodeploy is not available for Red Hat® Service Interconnect. Click **Next**.
3. Choose the **Autodeploy** option as a method to connect a gateway by clicking on the radio button.
4. Click **Select a VPC +** and select a vpc from the displayed list. Click **Select**.
5. Select the infrastructure group from the dropdown. Click **Next**.

6. Provide a gateway name. Choose a compute profile from the dropdown. Provide a label and description (optional).

7. Check the checkbox stating “By selecting the checkbox, you are acknowledging a VM, floating IP, and Route53 entry will be provisioned in your AWS account.”

8. Click **Submit**.

Deploying on-prem gateways in VMWare OCP

When installing a gateway for Red Hat® OpenShift® Container Platform (OCP) in an on-prem VMware vSphere environment, you use the Mesh CLI to manually deploy a Mesh edge gateway in a virtual machine (VM) next to the OCP instance.

Note: OCP supports multiple network plugins through the Container Network Interface (CNI). Currently, the only CNI that is supported by the gateway is Calico. By default, OCP uses the OpenShift SDN CNI which is not yet supported by Mesh.

Prerequisites

- Manually register the on-prem cloud, location, and OCP cluster with IBM Hybrid Cloud Mesh (Mesh).
- If you do not have the vCenter CLI `govc` command installed, see govc installation.

`govc` configuration

All `govc` commands need to receive a common set of parameters, for example, the URL of the vCenter server, username, password, etc. Export the following environment variables so you do not have to add them to every `govc` command.

```
export GOVC_URL="<url_to_vsphere_server>"
export GOVC_USERNAME="<username>"
export GOVC_PASSWORD="<password>"
export GOVC_TLS_CA_CERTS="<ca_certificate_of_the_server>"
export GOVC_DATACENTER="<virtual datacenter name>"
export GOVC_NETWORK="<network name>"
export GOVC_DATASTORE="<datastore name>"
```

For a complete list of the available `govc` environment variables, see its Usage section.

Creating the Mesh edge gateway resource and updating the cluster resource

Create the `gateway` resource (substituting a value for `< >`):

```
cat << EOM | palmctl create gateway -f -
name: <choose-any-name>
type: edge
EOM
```

Note the `resource_id` and `number` of the gateway in the output of the previous command and set them in variables for subsequent steps:

```
GW_RESOURCE_ID=<resource_id>
GW_NUMBER=<number>
```

Before associating the gateway with the cluster, create a Kubernetes secret. See Creating a Kubernetes secret.

Associate the gateway with the deployment environment (cluster instance) and enable discovery of namespaces and applications:

```
cat << EOM | palmctl patch cluster --cloud-name <cloud-resource-name> --name
<cluster-resource-name> -f -
gateway_id: $GW_RESOURCE_ID
auto_discover: true
unmanaged: false
EOM
```

Importing the Mesh edge gateway template into a vSphere library

Create a dedicated content library to store the gateway template in. (This will simplify managing multiple gateway template versions in the future.)

```
govc library.create "<library name>"
```

Note: This will store the library in the `GOVC_DATASTORE` that was previously specified.

The gateway template is an Open Virtualization Format (OVF) template which consists of three files that should be placed in the same directory:

- An `.ovf` file that has the VM description
- A `.vmdk` file for the VM disk
- A `.mf` file containing the checksums for the previous two files

Contact Mesh support to get the VMWare Mesh gateway VM image.

Use a naming convention for the base name of these files, for example `mesh-gw-<version>-template`, to enable managing multiple versions of the gateway in the future.

Import the gateway template into the content library previously created:

```
govc library.import -n="<library item name>" -m=true "<library name>" "<path to
gateway .ovf file>"
```

Notes:

- Although this command only requires the path to the `.ovf` file, it will automatically import all three files to the content library.
- The library name is analogous to a folder in a file system, and the library item name is analogous to a file within that folder.

Creating a VM from the gateway template

Deploy a VM from the template that is stored in the content library:

```
VM_NAME="<vm_name>"
govc library.deploy "<path_to_template>" "$VM_NAME"
```

Notes:

- The `<path_to_template>` is usually `/<library_name>/<template_name>`.
- The name of the VM is being saved in a variable for use in subsequent commands.
- The created VM will be in the `GOVC_DATACENTER` and will be in the powered off state.
- The default VM settings that are built in the gateway template will be used. For example, the VM will use 4 vCPU and 16GB RAM. These settings and other settings like networking configuration might need to be adapted to the execution environment. For more information, see VMware documentation: Deploy a Virtual Machine from a Template.

Booting up gateway VM and providing runtime configuration

The gateway VM expects some Mesh information to be provided at boot time so it can properly connect to the rest of the Mesh platform. Perform the steps in this section to gather the information required and boot the gateway VM with it.

Run these commands to set some shell variables for a subsequent step:

```
KAFKA_TOPIC_REQ="$(palmctl get gateway "$GW_RESOURCE_ID" | grep ' req_topic:' |
awk '{print $2}')"
KAFKA_TOPIC_REP="$(palmctl get gateway "$GW_RESOURCE_ID" | grep ' rsp_topic:' |
awk '{print $2}')"
KAFKA_TOPIC_DISCOVERY_REQ="$(palmctl get gateway "$GW_RESOURCE_ID" | grep
discovery_req_topic: | awk '{print $2}')"
KAFKA_TOPIC_DISCOVERY_REP="$(palmctl get gateway "$GW_RESOURCE_ID" | grep
discovery_resp_topic: | awk '{print $2}')"
KAFKA_TOPIC_HEALTH_REQ="$(palmctl get gateway "$GW_RESOURCE_ID" | grep
health_req_topic: | awk '{print $2}')"
KAFKA_TOPIC_HEALTH_REP="$(palmctl get gateway "$GW_RESOURCE_ID" | grep
health_rsp_topic: | awk '{print $2}')"
KAFKA_TOPIC_EVS="$(palmctl get gateway "$GW_RESOURCE_ID" | grep evs_topic: | awk
'{print $2}')"
```

Set your ssh public key in this variable, so you can ssh into the gateway VM after it is booted. **Note:** The gateway VM does not allow password-based logins, so providing an ssh public key is necessary to enable access to the gateway when troubleshooting is required.

```
YOUR_SSH_PUBLIC_KEY='<your ssh public key>'
```

Contact Mesh support to get these values and set them in these variables:

```
KAFKA_HOSTS='<comma-separated list of hosts>'
KAFKA_USERNAME='<kafka user>'
KAFKA_PASSWORD='<kafka password>'
OBS_TENANT_ID='<tenant ID for the observability stack>'
OBS_AUTH_CRED='<secret for remote communication to HUB>'
```

Run this command to create a file called `userdata.yaml` for the gateway VM:

```
cat << EOM > userdata.yaml
#cloud-config
write_files:
- encoding: text/plain
  content: |
    GWID=$GW_NUMBER
    KAFKA_HOSTS=$KAFKA_HOSTS
    KAFKA_USERNAME=$KAFKA_USERNAME
    KAFKA_PASSWORD=$KAFKA_PASSWORD
    KAFKA_PARTITIONS=1
    KAFKA_REPFACTOR=3
    KAFKA_TOPIC_EVS_TE=
    KAFKA_TOPIC_REQ=$KAFKA_TOPIC_REQ
    KAFKA_TOPIC_REP=$KAFKA_TOPIC_REP
    KAFKA_TOPIC_DISCOVERY_REQ=$KAFKA_TOPIC_DISCOVERY_REQ
    KAFKA_TOPIC_DISCOVERY_REP=$KAFKA_TOPIC_DISCOVERY_REP
    KAFKA_TOPIC_HEALTH_REQ=$KAFKA_TOPIC_HEALTH_REQ
    KAFKA_TOPIC_HEALTH_REP=$KAFKA_TOPIC_HEALTH_REP
    KAFKA_TOPIC_EVS=$KAFKA_TOPIC_EVS
    OBS_REMOTE_URL=https://gateway.multicloudmesh.ibm.com/thanos/api/v1/receive
    OBS_TENANT_ID=$OBS_TENANT_ID
    OBS_AUTH_CRED=$OBS_AUTH_CRED
    GW_SECRETS_PATH=/opt/palmetto/secrets
    GW_SECRETS_LOCAL_PATH=/opt/palmetto
  path: /etc/gw.conf
  permissions: '0640'

users:
  - name: palmetto
    ssh_authorized_keys:
      - ssh-rsa $YOUR_SSH_PUBLIC_KEY
EOM
```

Use the following commands to encode the file in the format that vSphere expects, modify the VM that was created in the previous section, and power it on:

```
USERDATA=$(gzip -c9 <userdata.yaml | base64 --wrap=0)
govc vm.change -vm "$VM_NAME" -e guestinfo.userdata="$USERDATA"
govc vm.change -vm "$VM_NAME" -e guestinfo.userdata.encoding=gzip+base64
govc vm.power -on "$VM_NAME"
```

Note: The Mesh edge gateway has two requirements for the IP addresses it is assigned when the VM is booted up:

- A private IP address through which it can reach the OCP cluster and its workloads.
- An internet-facing (public) IP address through which it can reach the other Mesh gateways and the Mesh SaaS.

Run the following commands to store the private and public IP addresses that are assigned to the gateway VM in variables:

```
GW_PRIVATE_IP=$(govc vm.ip "$VM_NAME")
GW_PUBLIC_IP=<internet-facing IP for the VM>
```

Network configuration

The steps in the following sub-sections cannot be performed using **govc** commands. Instead, they need to be performed on the vSphere datacenter. These steps are highly dependent on your specific setup, so these sections only provide high-level instructions.

Adding static routes for the gateway VM

Forward traffic from the OCP workloads in the VMware datacenter to the gateway by adding a static route to the network which the workloads are attached to. This route must target traffic whose destination IP belongs to Classless Inter-Domain Routing (CIDR) `100.64.0.0/10`, and the route must point to the `GW_PRIVATE_IP`.

There are different ways to achieve this:

- One method is to add this route to each OCP worker node where the workload pods are running, using a command like: `ip route add 100.64.0.0/10 via $GW_PRIVATE_IP`. Note that this command needs to be reapplied on every boot of each worker node. There are different methods to make this persistent, depending on the OS of the worker node. Although this method works, it requires additional configuration work every time a new worker node is added.
- Another method is to add the route on the routers (virtual or physical) that are a attached to the networks where the OCP worker nodes are. The method to do this is dependent on the networking of the OCP cluster.

If the OCP pod and service CIDRs of the cluster are not directly reachable from the gateway, you need to add routes for those.

Configuring OCP IP preservation

In order to apply Mesh application and service connection policies, the gateway must be able to identify the applications and services that are trying to communicate. The default configuration for the OCP Calico CNI is to source NAT (masquerade) traffic directed to IP addresses outside of the OCP internal network. This needs to be disabled for traffic flowing to CIDRs used in the Mesh overlay `100.64.0.0/10`. This can be achieved by creating a new IPPool configuration with `disabled=true`. See Configure outgoing NAT for details.

Opening firewall ports

If there are firewalls in place, some ports on them need to be opened for the gateway. For traffic between the gateway and the internet:

- Egress traffic will be going out of the gateway on UDP ports 6081 and 4500 for the (GENEVE) and secure (IPsec) encapsulations, respectively. Also, the gateway will attempt to connect to the hosts specified in the `KAFKA_HOSTS` variable in the `userdata.yaml` file (typically port TCP 9092 or 9093). The gateway might also establish connections on TCP port 53 for DNS transfers.
- For ingress traffic, the gateway needs UDP ports 6081 and 4500 open so that it can setup the tunnels to other Mesh gateways for the (GENEVE) and secure (IPsec) encapsulations, respectively. If an external SNMP manager is used to access the gateway's MIB, UDP port 161 might need to be opened, depending on the location of the SNMP agent.
- The gateway VM accepts SSH connections on port TCP 7777. Ensure that this port is also open.

For traffic between the gateway and the OCP cluster:

- Allow port UDP 53 (DNS).
- Allow traffic originating from the gateway and targeting the OCP cluster's API endpoint (master nodes): srcIP=GW_PRIVATE_IP, dstIP=`<any_master_node>`, dstPort=
- Allow traffic to and from the application and service global IP range: src_CIDR=`100.64.0.0/10`, dstIP=

Updating the gateway resource with IP addresses

Update the Mesh `gateway` resource with the IP addresses from the previous steps.

```
cat << EOM | palmctl patch gateway --id $GW_RESOURCE_ID -f -
public_ip: $GW_PUBLIC_IP
private_ip: $GW_PRIVATE_IP
EOM
```

Note: This will automatically change the gateway status to `operational` and cause Mesh to start managing it.

What's next

See Discovering cluster namespaces and application.

Monitoring gateways

To monitor the operational status of your Mesh gateways, run the following command:

```
palmctl get gateways | grep -E ' *(name|status):'
```

If you have installed the yq command, run the following command to output the **name** and **status** of each gateway in a single line:

```
palmctl get gateways | yq '.gateways[] | "name: " + .name + ", status: " +
.status'
```

Updating gateways

When a new gateway VM image is available, you will be notified via an email from **mesh-support@ibm.com**. You may update a gateway by having IBM Hybrid Cloud Mesh (Mesh) delete it and create it:

1. Find the gateway deployment ID of the gateway to be updated:

   ```
   palmctl get gwdeployments --cloud-name <cloud-name> --vpc-name <vpc-name>
   ```

   Set the **gw_deploy_id**, **gw_resource_id**, and **compute_profile** field values of the gateway deployment you want to update in variables for use in subsequent commands:

   ```
   GW_DEPLOY_ID="<gw_deploy_id>"
   GW_RESOURCE_ID="<gw_resource_id>"
   COMPUTE_PROFILE="<compute_profile>"
   ```

2. Remove the gateway deployment:

   ```
   palmctl delete gwdeployment --cloud-name <cloud-name> --vpc-name <vpc-name> -
   -gw-deployment-id $GW_DEPLOY_ID
   ```

3. Deploy the gateway with the latest image:

   ```
   cat << EOM | palmctl create gwdeployment --cloud-name <cloud-name> --vpc-name
   <vpc-name> -f -
   gw_resource_id: $GW_RESOURCE_ID
   compute_profile: $COMPUTE_PROFILE
   EOM
   ```

Removing a gateway

When a gateway is no longer needed, it can be removed by IBM Hybrid Cloud Mesh (Mesh). This would be the case when, for example, applications that needed connecting were moved out of a deployment environment.

1. Find the gateway deployment ID of the gateway to be removed:

   ```
   palmctl get gwdeployments --cloud-name <cloud-name> --vpc-name <vpc-name>
   ```

   Set the **gw_deploy_id** and **gw_resource_id** field values of the gateway deployment you want to update in variables for use in subsequent commands:

   ```
   GW_DEPLOY_ID="<gw_deploy_id>"
   GW_RESOURCE_ID="<gw_resource_id>"
   ```

2. Remove the gateway deployment:

   ```
   palmctl delete gwdeployment --cloud-name <cloud-name> --vpc-name <vpc-name> -
   -gw-deployment-id $GW_DEPLOY_ID
   ```

   Note: When this gateway was originally deployed, a route was created in the VPC to direct **100.64.0.0/10** network traffic to the gateway. If a route for **100.64.0.0/10** existed before that time, it was overwritten. Deleting this gateway deployment now does not restore the original route, instead it removes the **100.64.0.0/10** route. You must restore the route manually. This will be fixed in a future release.

3. Remove the gateway resource:

   ```
   palmctl delete gateway $GW_RESOURCE_ID
   ```

Deploying gateways in AWS

Use the Mesh CLI to auto-deploy gateways in AWS EKS.

A Mesh gateway must be deployed for each deployment environment that is running applications and services you want Mesh to provide connectivity for.

Prerequisites

- Create a secret for the cloud credentials and set the secret in the cloud resource, as described in Discovering cloud infrastructure.

If you are setting up an EKS cluster, you need to set **auto_discover: true** in the **cloud** resource in Discovering cloud infrastructure. **Manual registration of cloud components for EKS is currently not supported**.

- Register the VPC and cluster resources either by autodiscovery or manually registering the resources.

Creating the gateway resource

Create the **gateway** resource (substituting a value for **< >**):

```
cat << EOM | palmctl create gateway -f -
name: <choose-any-name>
type: <edge or waypoint>
EOM
```

Note the **resource_id** of the gateway in the output of the previous command and set it in a variable for subsequent steps:

```
GW_RESOURCE_ID=<resource_id>
```

Get the cloud resource name by running `palmctl get clouds`. Note the `name` of the cloud that the gateway will be in, and save it in a variable for subsequent steps:

```
CLOUD_RESOURCE_NAME=<name>
```

Get the VPC resource name by running `palmctl get vpcs --cloud-name "$CLOUD_RESOURCE_NAME"`. Note the `name` of the VPC that the gateway will be in, and save it in a variable for subsequent steps:

```
VPC_RESOURCE_NAME=<name>
```

If this is an `edge` gateway (not a `waypoint` gateway), associate the gateway with the deployment environment (cluster instance) and enable discovery of namespaces and applications:

```
cat << EOM | palmctl patch cluster --cloud-name "$CLOUD_RESOURCE_NAME" --name
<cluster-resource-name> -f -
gateway_id: $GW_RESOURCE_ID
auto_discover: true
unmanaged: false
EOM
```

Deploying the gateway

The following command initiates the deployment and configuration of the gateway:

```
cat << EOM | palmctl create gwdeployment --cloud-name "$CLOUD_RESOURCE_NAME" --
vpc-name "$VPC_RESOURCE_NAME" -f -
gw_resource_id: $GW_RESOURCE_ID
compute_profile: <compute-profile>
EOM
```

Set `compute_profile` to one of the values in the first column of the following table, depending on the application network bandwidth required. If you do not specify `compute_profile`, `small` is used.

Compute Profile	Vendor Instance	vCPU	Mem. (GiB)	Network Bandwidth (Gbps)	Comment
`small`	`c5n.xlarge`	4	10.5	up to 25 Gbps	Minimum required to run the gateway. This is the default.
`medium`	`c5n.2xlarge`	8	21	up to 25 Gbps	Moderate cost
`large`	`c5n.4xlarge`	16	41	up to 25 Gbps	Optimized for networking

When the gateway is deployed, Mesh performs the following steps:

- A new public subnet is created where the GW VM will be deployed.
- A strict security group (firewall) is created to ensure only the relevant ports are exposed to the Internet. See the list of ports that are opened.
- The gateway VM is deployed and assigned the security group.
- The gateway VM calls home to Mesh, and from that point it is managed and monitored exclusively by Mesh.
- A new outbound endpoint in the Route53 resolver is configured to forward the DNS queries for the subdomain `palmetto.ibm.com` to the gateway.

- For every workload route table that has connected subnets, a route for `100.64.0.0/10` is added via the gateway.

The output of the command will include a field called **`gw_deploy_id`**. Set its value in this variable for subsequent commands:

```
gw_deploy_id=<gw deploy id>
```

Waiting for the gateway deployment to complete

In AWS, the deployment of a gateway can take up to 30 minutes to add subnet routes for the DNS resolution of **`palmetto.ibm.com`**. Check the status of the deployment by running:

```
palmctl get gwdeployment --cloud-name <cloud-name> --vpc-name <vpc-name> --gw-
deployment-id $gw_deploy_id
```

Once the **`status`** field of the command output is **`operational`**, the gateway is ready to be used to create policies to connect applications. From this point on, Mesh will:

- Actively monitor any change in the portion of the VPC networking configuration that it manages, including the security group configuration.
- Watch for an unintended change in the gateway configuration, for example, due to a misconfiguration through the cloud REST API or UI. In this case, the gateway configuration will be restored automatically by Mesh.

If you accidentally delete any resource, such as the gateway instance, routes, subnets, or security group through the AWS console or CLI, you need to redeploy the gateway. You can redeploy the gateway by deleting the existing gateway deployment and creating a new gateway deployment. For more information about how to delete a gateway deployment, see Removing gateways.

What's Next

See Discovering cluster namespaces and application.

Deploying gateways in IBM Cloud

Use the Mesh CLI to auto-deploy gateways in IBM Cloud®.

An IBM Hybrid Cloud Mesh (Mesh) gateway needs to be deployed for each deployment environment that is running applications and services you want Mesh to provide connectivity for. This document describes how to deploy a gateway for a cluster deployment environment.

Prerequisites

- As part of gateway deployment, an IP spoofing check is enabled for the gateway VM to ensure that traffic that is coming from the network interface includes appropriate addressing. Enable the IP Spoofing Operator role in your Identity and Access Management (IAM) authorization to update the network interface to enable or disable the IP spoofing check. For more information, see IP spoofing checks
- Create a secret for the cloud credentials and set it in the **`cloud`** resource, as described in Discovering cloud infrastructure.

If you are setting up an IKS/ROKS cluster, you need to set **auto_discover: true** in the **cloud** resource in Discovering cloud infrastructure. **Manual registration of cloud components for IKS/ROKS is currently not supported**.

- Register the cluster resource and encompassing VPC resource by either discovering them, or manually registering them.
- Add a Cloud Object Storage bucket URL in your IBM Cloud account. The serviceID and the API key that you generated for the Mesh secrets must have permissions to access your Cloud Object Storage bucket service instance. For more information, see Adding Cloud Object Storage bucket URL.

Creating the gateway resource

Create the **gateway** resource (substituting a value for **< >**):

```
cat << EOM | palmctl create gateway -f -
name: <choose-any-name>
type: <edge or waypoint>
EOM
```

Note the **resource_id** of the gateway in the output of the previous command and set it in a variable for subsequent steps:

```
GW_RESOURCE_ID=<resource_id>
```

Get the cloud resource name by running **palmctl get clouds**. Note the **name** of the cloud that the gateway will be in, and save it in a variable for subsequent steps:

```
CLOUD_RESOURCE_NAME=<name>
```

Get the VPC resource name by running **palmctl get vpcs --cloud-name "$CLOUD_RESOURCE_NAME"**. Note the **name** of the VPC that the gateway will be in, and save it in a variable for subsequent steps:

```
VPC_RESOURCE_NAME=<name>
```

If this is an **edge** gateway (not a **waypoint** gateway), associate the gateway with the deployment environment (cluster instance) and enable discovery of namespaces and applications:

```
cat << EOM | palmctl patch cluster --cloud-name "$CLOUD_RESOURCE_NAME" --name 
<cluster-resource-name> -f -
gateway_id: $GW_RESOURCE_ID
auto_discover: true
unmanaged: false
EOM
```

Deploying the gateway

The following command initiates the deployment and configuration of the gateway:

```
cat << EOM | palmctl create gwdeployment --cloud-name "$CLOUD_RESOURCE_NAME" --
vpc-name "$VPC_RESOURCE_NAME" -f -
gw_resource_id: $GW_RESOURCE_ID
compute_profile: <compute-profile>
EOM
```

Set **compute_profile** to one of the values in the first column of the following table, depending on the application network bandwidth required. If you do not specify **compute_profile**, **small** is used.

Compute Profile	Instance	vCPU	Mem. (GiB)	Network Bandwidth (Gbps)	Comment
`small`	`cx2d-4x8`	4	8	up to 8 Gbps	Minimum required to run the gateway. This is the default.
`medium`	`cx2d-8x16`	8	16	up to 16 Gbps	Moderate cost
`large`	`cx2d-16x32`	16	32	up to 32 Gbps	Optimized for networking

When the gateway is deployed, Mesh performs the following steps:

- A new public subnet is created where the GW VM will be deployed.
- A strict security group (firewall) is created to ensure only the relevant ports are exposed to the Internet. See the list of ports that are opened.
- The gateway VM is deployed and assigned the security group.
- The gateway VM calls home to Mesh, and from that point it is managed and monitored exclusively by Mesh.
- The k8s coredns is configured to forward the DNS queries for the subdomain palmetto.ibm.com to the gateway.
- For every workload route table that has connected subnets, a route for `100.64.0.0/10` is added via the gateway.

The output of the command will include a field called `gw_deploy_id`. Set its value in this variable for subsequent commands:

```
gw_deploy_id=<gw deploy id>
```

Waiting for the gateway deployment to complete

The deployment of a gateway can take several minutes. Check the status of the deployment by running:

```
palmctl get gwdeployment --cloud-name <cloud-name> --vpc-name <vpc-name> --gw-deployment-id $gw_deploy_id
```

Once the `status` field of the command output is `operational`, the gateway is ready to be used to create policies to connect applications. From this point on, Mesh will:

- Actively monitor any change in the portion of the VPC networking configuration that it manages, including the security group configuration.
- Watch for an unintended change in the gateway configuration, for example, due to a misconfiguration through the cloud REST API or UI. In this case, the gateway configuration will be restored automatically by Mesh.

If you accidentally delete any resource, such as the gateway instance, routes, subnets, or security group through the IBM® cloud console or CLI, you need to redeploy the gateway. You can redeploy the gateway by deleting the existing gateway deployment and creating a new gateway deployment. For more information about how to delete a gateway deployment, see Removing gateways.

What's Next

See Discovering cluster namespaces and application.

Adding a Cloud Object Storage bucket URL

Add a Cloud Object Storage bucket URL in IBM Cloud® to deploy the Mesh gateways.

Complete the following steps to add a Cloud Object Storage bucket URL in IBM Cloud:

1. Create an IBM Cloud Object Storage service instance, and then add a cross-region bucket to store your data. For more information, see Getting started with IBM Cloud Object Storage.

2. Copy the Cloud Object Storage bucket URL from the bucket details page. You can store your Cloud Object Storage bucket URL in an environment variable for subsequent steps.

   Run the following command to get the Cloud Object Storage bucket URL:

   ```
   export IBM_COS_BUCKET_URL=cos://<your-region-name>/<your-bucket-name>/
   ```

3. Create an access group with the specified permissions and generate a serviceID with API key to use as a secret. The serviceID and the API key that you generated must have permissions to access your Cloud Object Storage bucket service instance with the following roles:

   - Manager
   - RC KeyManager
   - Viewer

   For more information about how to grant permissions, see Granting user permissions to a user or service ID.

4. Grant access to IBM Cloud Object Storage to import and export images to IBM Cloud VPC. For more information, see Creating an authorization.

5. Configure Mesh to use the Cloud Object Storage bucket URL that you got in step 2 for uploading the Axon gateway images. Run a command like this:

   ```
   cat <<EOF | palmctl create tenant-configuration -f -
   ibm_cos_bucket_url: ${IBM_COS_BUCKET_URL}
   EOF
   ```

6. You can verify the tenant configuration by running the following command:

   ```
   palmctl get tenant-configuration
   ```

7. Run the following commands to update, patch, or delete the Cloud Object Storage bucket URL.

   - To update, run a command like this:

     ```
     cat <<EOF | palmctl update tenant-configuration -f -
     ibm_cos_bucket_url: cos://<your-region-name>/<your-bucket-name>/
     EOF
     ```

   - To patch, run a command like this:

     ```
     cat <<EOF | palmctl patch tenant-configuration -f -
     ibm_cos_bucket_url: cos://<your-region-name>/<your-bucket-name>/
     EOF
     ```

   - To delete, run a command like this:

     ```
     palmctl delete tenant-configuration
     ```

Technology previews

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

For more information about Red Hat® Service Interconnect, see Working with Red Hat® Service Interconnect.

Working with Red Hat Service Interconnect

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Red Hat Service Interconnect simplifies application connectivity across the hybrid cloud and enables your applications to span multiple cloud providers, data centers, and regions. Unlike traditional means of interconnectivity, such as VPNs combined with complex firewall rules, development teams can easily create interconnections without elevated privileges and deliver protected links without compromising the organization's security or data.

Applications and services across your environment can communicate with each other using Red Hat Service Interconnect as if they are all running in the same site.

The primary use case for Red Hat Service Interconnect is to connect Kubernetes namespaces across distributed Kubernetes deployments. The Red Hat Service Interconnect router is deployed in the same namespace as the applications and service and provides networking services just for the namespace that it lives in.

Use the Red Hat Service Interconnect architecture to connect gateways into isolated sets of interconnected gateways. Then, expose a service to all the namespaces that belong to a single interconnected gateway set.

For more information about Red Hat Service Interconnect, see Red Hat Service Interconnect.

Network segments in IBM Hybrid Cloud Mesh

IBM Hybrid Cloud Mesh provides a management layer for Red Hat Service Interconnect to give customers access to Red Hat Service Interconnect technology.

In IBM Hybrid Cloud Mesh, the sets of interconnected gateways are called network segments. A network segment supports the creation of gateway interconnects and prevents interconnections with Red Hat Service Interconnect gateways that are not in the same network segment.

When you define namespaces, applications, and policies, including their child resources, you assign them to a specific network segment. When a service in a network segment is exposed, any application in the network segment can communicate with that service.

To enable the applications in a network segment to communicate with a service in the same network segment, you create a Mesh policy. The `from` side of the policy is set to the network segment that contains the applications and the `to` side is set to a service in the same network segment.

What to do next

Complete the following steps to get started with Red Hat Service Interconnect:

1. Install the Open Horizon agent. The agent manages the lifecycle of Red Hat Service Interconnect gateways. See Installing the Open Horizon agent.

2. Deploy your Red Hat Service Interconnect gateways by using the Mesh console or the CLI:

    1. Create a network segment.
    2. Create your Service Interconnect edge gateways and connect the gateways by using a remote connection.
    3. Create connection policies so that service requests can flow over your Service Interconnect edge gateways.

    See Deploying gateways with the console and Deploying gateways with the CLI.

Installing the Open Horizon agent

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

IBM Hybrid Cloud Mesh uses Open Horizon to deploy containers to edge clusters. When the Open Horizon agent is installed, the agent manages the lifecycle of Red Hat Service Interconnect gateways.

Before you can deploy a Service Interconnect edge gateway in a Kubernetes namespace, you must install the Open Horizon agent in that namespace. The procedure for installing the Open Horizon agent is slightly different for each Kubernetes environment.

Prerequisites

1. Ensure that the host machine from which you start the agent installation meets the following requirements:

    - The host is a Linux® machine or virtual machine with the AMD64 architecture.
    - Docker or Podman is installed on the host.
    - The host can access the Kubernetes cluster with the `kubectl` CLI or the `oc` CLI for OpenShift®. You log in to the cluster as a user with administrative privileges.

2. To get the installation package for the Open Horizon agent, install and configure the Mesh CLI on the host machine.

3. Configure a Kubernetes storage class that can be used to satisfy the Open Horizon agent's persistent volume claim. The volume must be immediately available, and the agent must have read and write access to the volume.

4. Configure a container registry that is available and accessible from the cluster. The agent container image for the cluster is stored in the registry so that the image is always available.

    The installation script for the Open Horizon agent downloads the agent from Mesh and pushes the agent into the registry. The Kubernetes deployment for the Open Horizon agent is configured to point to the registry location.

    The installation instructions assume that a registry outside the cluster is used to hold the Open Horizon agent.

To configure your registry, use the following example guidelines for the different container registries:

- IBM Cloud Container Registry
  - Create a namespace in a specific region or a global namespace.
  - Set `EDGE_CLUSTER_REGISTRY_USERNAME` to `iamapikey`.
  - Create an IAM API key and set `EDGE_CLUSTER_REGISTRY_TOKEN` to the key value.
  - Set `IMAGE_ON_EDGE_CLUSTER_REGISTRY` to a value like `us.icr.io/<your_namespace>/amd64_anax_k8s`.
- AWS Elastic Container Registry
  - Create two repositories, `amd64_anax_k8s` and `amd64_auto-upgrade-cronjob_k8s`.
  - Set `EDGE_CLUSTER_REGISTRY_USERNAME` to `AWS`.
  - Use this AWS CLI command to get the value that you use for `EDGE_CLUSTER_REGISTRY_TOKEN`: `aws ecr get-login-password --region <your_region>`
  - Set `IMAGE_ON_EDGE_CLUSTER_REGISTRY` to a value like `<your_aws_account_num>.dkr.ecr.<your_region>.amazonaws.com/amd64_anax_k8s`.
- Docker and Quay.io container registries
  - Create two repositories, `amd64_anax_k8s` and `amd64_auto_upgrade_cronjob_k8s`.
  - Set `EDGE_CLUSTER_REGISTRY_USERNAME` to your Docker or Quay.io username.
  - Set `EDGE_CLUSTER_REGISTRY_TOKEN` to your container registry password.
  - Set `IMAGE_ON_EDGE_CLUSTER_REGISTRY` to a value like `quay.io/<your_username>/amd64_anax_k8s`.

5. For each supported Kubernetes environment, complete the following instructions to set the storage class and the environment variables for the container registry:

Red Hat® OpenShift® Kubernetes Service (ROKS)

Set the storage class for the Kubernetes environment. For example:

```
export EDGE_CLUSTER_STORAGE_CLASS=ibmc-vpc-block-10iops-tier
```

Set these environment variables to direct the agent installer where to store the agent container:

```
export USE_EDGE_CLUSTER_REGISTRY="false"
export IMAGE_ON_EDGE_CLUSTER_REGISTRY=<IBM container registry URL>
export EDGE_CLUSTER_REGISTRY_USERNAME=<user that can access the registry>
export EDGE_CLUSTER_REGISTRY_TOKEN=<credential to access the registry>
```

AWS Elastic Kubernetes Service (EKS)

Set the storage class for the Kubernetes environment. For example:

```
export EDGE_CLUSTER_STORAGE_CLASS=ebs-sc
```

To use the **ebs-sc** storage class, ensure that the Amazon Elastic Block Store (EBS) driver is installed. See Amazon EBS CSI driver.

Set these environment variables to direct the agent installer where to store the agent container:

```
export USE_EDGE_CLUSTER_REGISTRY="false"
export IMAGE_ON_EDGE_CLUSTER_REGISTRY=<AWS container registry URL>
export EDGE_CLUSTER_REGISTRY_USERNAME=<user that can access the registry>
export EDGE_CLUSTER_REGISTRY_TOKEN=<credential to access the registry>
```

Red Hat OpenShift Service on AWS (ROSA)

Set the storage class for the Kubernetes environment. For example:

```
export EDGE_CLUSTER_STORAGE_CLASS=gp2
```

Set these environment variables to direct the agent installer where to store the agent container:

```
export USE_EDGE_CLUSTER_REGISTRY="false"
export IMAGE_ON_EDGE_CLUSTER_REGISTRY=<AWS container registry URL>
export EDGE_CLUSTER_REGISTRY_USERNAME=<user that can access the registry>
export EDGE_CLUSTER_REGISTRY_TOKEN=<credential to access the registry>
```

Red Hat OpenShift Container Platform

For this Kubernetes environment, the available storage classes depend on how Kubernetes is configured. Work with your Kubernetes administrator to find a suitable storage class.

```
export EDGE_CLUSTER_STORAGE_CLASS=<storage class>
```

Set these environment variables to direct the agent installer where to store the agent container:

```
export USE_EDGE_CLUSTER_REGISTRY="false"
export IMAGE_ON_EDGE_CLUSTER_REGISTRY=<container registry URL>
export EDGE_CLUSTER_REGISTRY_USERNAME=<user that can access the registry>
export EDGE_CLUSTER_REGISTRY_TOKEN=<credential to access the registry>
```

Installing the Open Horizon agent

Complete the following steps:

1. Choose a meaningful name for your Open Horizon agent, and set the `HZN_NODE_ID` environment variable to that name.

   Ensure that the name is meaningful enough so that the name can be used to identify the cluster and the namespace where the agent is installed. The name must be unique within your Mesh tenant. For example:

   ```
   export HZN_NODE_ID=sales-cluster-frontend-ns-agent
   ```

   When you configure a Service Interconnect edge gateway, you choose the Open Horizon agent that manages the gateway.

2. Set the Kubernetes namespace where you want the Service Interconnect edge gateway to be installed.

   ```
   export AGENT_NAMESPACE=frontend-ns
   ```

3. Disable automatic upgrades for the agent. The automatic upgrade of the Open Horizon agent is not supported in the technology preview for Red Hat® Service Interconnect.

   ```
   export ENABLE_AUTO_UPGRADE_CRONJOB=false
   ```

4. Download the installation package for the Open Horizon agent from Mesh and extract the package:

   ```
   palmctl get openhorizon
   tar -xvzf openhorizon-agent-install-files.tar.gz
   chmod 755 agent-install.sh
   ```

5. Open the `agent-install.cfg` file and copy the `HZN_EXCHANGE_USER_AUTH` variable. Set the variable in your shell. For example, `agent-install.cfg` might contain the following settings:

```
HZN_ORG_ID=...
HZN_EXCHANGE_USER_AUTH=someuser:password
HZN_EXCHANGE_URL=...
```

6. Set the **`HZN_EXCHANGE_USER_AUTH`** variable in your shell:

   ```
   HZN_EXCHANGE_USER_AUTH=someuser:password
   ```

   If the password contains special characters, you must enclose the value of the variable in single quotation marks. For example, **`HZN_EXCHANGE_USER_AUTH='someuser:password'`**

7. Run the agent installation script:

   ```
   sudo -s -E ./agent-install.sh -D cluster -i 'css:' -u $HZN_EXCHANGE_USER_AUTH
   --namespace $AGENT_NAMESPACE \
     --namespace-scoped -k ./agent-install.cfg
   ```

8. Verify that the Open Horizon agent is installed and running:

   ```
   kubectl get pods -n $AGENT_NAMESPACE
   ```

   If the agent pod is running, the output looks like this:

   ```
   NAME                        READY   STATUS    RESTARTS   AGE
   agent-7459dc47dc-mjf8w      1/1     Running   0          5d19h
   ```

   When the installation script completes successfully, you can select the Open Horizon agent when you are deploying the Service Interconnect edge gateway.

What to do next

Deploy your Red Hat Service Interconnect gateways. See Deploying gateways with the console and Deploying gateways with the CLI.

Deploying Service Interconnect edge gateways with the Mesh console

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Create a network segment

Mesh enables you to create a new network segment for the Red Hat Service Interconnect gateway only. Follow these steps to create a new network segment for Red Hat Service Interconnect gateway:

1. Go to **Network segment** page on the Mesh console.
2. Click **Create network segment**.
3. Provide name, and select an infrastructure group from the dropdown. Tag and description are optional.
4. Click **Create**

Success notification appears once the network segment is created. You can view the registered network segment listed on the table on **Network segment** page. Now you can choose the created network segment while registering an application, registering a namespace and creating a policy.

Register namespace with connect a gateway using the agent

Follow these steps to register a namespace with **connect a gateway from agent** option:

1. Go to the **Deployment environments** page. Choose a deployment environment of type cluster from the displayed list.
2. Scroll down on the **Deployment environment details** page and click **Register namespace +**.
3. Choose a network segment from the **Network segment** dropdown or you can choose to create a new network segment, by clicking on **Create a network segment** option from the dropdown list.
4. Click **Connect to gateway +**.
5. Select **Create a gateway from agent** option.
6. Select an unassigned agent from the agent dropdown. The **Namespace** and **Gateway** fields are auto-populated as per the selection of agent. You can edit the gateway name.
7. Click **Create**. Success notification for gateway creation is displayed.
8. Provide labels and description (optional) for the namespace. Click **Register**.

Success notification for namespace creation is displayed.

Create a Red Hat Service Interconnect gateway

Prerequisite : Before creating a Red Hat Service Interconnect gateway through the Mesh console, install the Open Horizon agent. See Installing the Open Horizon agent.

Follow these steps to create a Red Hat Service Interconnect gateway through Mesh console:

1. Go to **Gateway** page on the Mesh console.
2. Click **Register gateway**
3. Select the Red Hat Service Interconnect as gateway type by clicking on it.
4. Select the network segment from the dropdown. You can choose to create a new network segment by clicking on **Create a network segment +** option from the dropdown.
5. Select the linked agent name from the **Linked agent** dropdown.
6. Click on **Select a cluster+** to select the agent deployment cluster. Select a cluster from the displayed list of clusters on Cluster dialogbox, by clicking the radio button next to the option.
7. Select the infrastructure group and click **Next**.
8. Provide label and description (optional) and click **Submit**.

Connecting two Red Hat Service Interconnect gateways with a remote connection

Follow these steps to connect two gateways with a remote connection:

1. Go to **Gateways** page.
2. Click a Red Hat Service Interconnect gateway displayed on the list on **Gateways** page.
3. Scroll down on the gateway details page and click **connect connection +**.
4. Provide a value for the **Link metric**. By default, it is 1.
5. Select the second gateway from the displayed list and click on **Create**.

Success notification for remote connection between the two gateways is displayed.

Note: To connect two gateways using remote connections, both gateways should be associated with the same network segment.

Register a service and a service endpoint

To create the connection policy, you need a service and a service endpoint. If the application is not exposed by a service, complete the following steps to register a service and a service endpoint.

If the application is exposed by a service, the service and service endpoint are automatically registered during Mesh discovery.

1. On the Applications page, click the application for which you want to add a service.
2. Click **Register service +** in the Services section.
3. Add the service name.
4. Click `TCP` in the `Protocol` list and enter the port number for the application, for example `8080`. Then, click **Add** to add the port to the service.
5. Click **Register**.

A message is shown when the service is registered successfully.

Now, use the following steps to register the service endpoint:

1. On the Applications page, click the application for which you want to add a service endpoint.
2. In the Application deployments section, click the deployment that you want to update.
3. Click **Register service endpoint** in the Service endpoints section.
4. In the `Select a service` list, click the service that you registered in the previous step. If you do not have a specific local IP address, you can enter any valid value in the `Local IP` field, for example, `127.0.0.1`.
5. Click **Register +**.

A message is shown when the service endpoint is registered successfully for the deployment.

Creating a policy from service to the network segment

Follow these steps to create a policy from a service to the network segment:

1. Go to **Create access policies** page.
2. Click **Create policy**.
3. Provide policy name. Select a Service Interconnect® network segment from the **Network Segment** dropdown.
4. Provide label and description (optional). Under the **To** section **click on view all options +**.
5. Select the service to be connected and click **Save**.
6. Click **Create**.

Success notification for policy creation is displayed.

Note: The service must be in the same network segment as the policy.

Deploying Service Interconnect edge gateways with the CLI

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Deploy your Service Interconnect edge gateways in the Mesh CLI:

1. Create a network segment. See Creating Mesh network segments.
2. Create your Service Interconnect edge gateways and connect the gateways by using a remote connection. See Creating and connecting Service Interconnect edge gateways.
3. Create connection policies so that service requests can flow over your Service Interconnect edge gateways. See Creating connection policies.

Creating Mesh network segments

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Before you configure your Red Hat Service Interconnect gateways, you must create a Mesh network segment to support the interconnections between the Service Interconnect edge gateways.

For more information about network segments, see Working with Red Hat Service Interconnect.

Create the network segment

Run the following Mesh CLI command:

```
cat << EOM | palmctl create networksegment -f -
name: <network-segment-name>
compatibility_set: RHSI
EOM
```

`<network-segment-name>` must be unique.

`compatibility_set` must be set to `RHSI`.

What to do next

Create the Service Interconnect edge gateways and configure the connection between the gateways. See Creating and connecting Red Hat Service Interconnect gateways.

Creating and connecting Service Interconnect edge gateways

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Create your Red Hat Service Interconnect gateways in Mesh and connect the gateways by using a remote connection.

Prerequisites

1. Install an Open Horizon agent to manage the gateway. See Installing the Open Horizon agent.
2. Install the Mesh CLI. See Installing and configuring the CLI.
3. Create or identify a Mesh managed namespace that represents the Kubernetes namespace where you want to deploy the gateway.
4. Create or identify a Mesh network segment that contains the managed namespace. For more information about creating a network segment, see Creating network segments.

Create the gateway

Use the **'palmctl'** CLI to complete the following steps:

1. Get the ID of the cloud where you are deploying the gateway:

   ```
   palmctl get clouds
   ```

   The cloud ID is shown in the command output.

2. Create the Red Hat Service Interconnect gateway:

   ```
   cat << EOM | palmctl create gateway -f -
   name: <gateway-name>
   open_horizon_node_id: <open-horizon-agent-id>
   type: edge
   subtype: RHSI-EDGE
   number: 1
   cloud_id: <cloud-id>
   EOM
   ```

   `<gateway-name>` must be unique.

Connect two gateways

To connect two Service Interconnect edge gateways, you need to create a Mesh remote connection. The gateways that you are connecting must be associated with the same network segment.

A Mesh remote connection is unidirectional and has a source gateway and a destination gateway. However, Service Interconnect edge gateways are bidirectional. You do not need to create a Mesh remote connection in both directions.

To create a remote connection, you need the name of the source gateway and the ID of the destination gateway.

Complete the following steps:

1. Get the ID of the destination gateway:

   ```
   palmctl get gateways --cloud-name <cloud-name>
   ```

   `<cloud-name>` is the name of the Mesh cloud where the gateways are defined.
   The ID of the destination gateway is shown in the `resource_id` field in the output.

2. Create the remote connection:

   ```
   cat << EOM | palmctl create remoteconnection --gateway-name=<gateway-name> -f
   -
   name: <connection-name>
   remote_gw_id: <destination-gateway-resource-id>
   EOM
   ```

<gateway-name> is the name of the source gateway.
<connection-name> must be unique.

To establish communication between the gateways, Mesh requests a link secret from the destination gateway and deploys it on the source gateway.

What to do next

Create a Mesh connection policy so that service requests can flow over your Service Interconnect edge gateway. See Creating connection policies.

Creating connection policies

Red Hat® Service Interconnect® is one of the embedded gateways delivered by IBM Hybrid Cloud Mesh. This embedded gateway is a Tech Preview and thus not covered by any SLA and IBM Support.

Create connection policies in Mesh so that service requests can flow over your Red Hat Service Interconnect gateways. A connection policy defines which applications can use each service and enables connectivity between the applications and services that are referenced in the policy.

Applications and services in your Kubernetes namespace are discovered when the Service Interconnect edge gateway is deployed. For service requests to be routed over the Service Interconnect edge network, create a connection policy that allows requests for the service within the network segment.

Prerequisites

To create the connection policy, you need a service and a service endpoint. If the application is not exposed by a service, complete the following steps to register a service and a service endpoint.

If the application is exposed by a service, the service and service endpoint are automatically registered during Mesh discovery.

1. Get the application name:

   ```
   palmctl get applications
   ```

2. Create the service:

   ```
   cat << EOM | palmctl create service --application-name <application-name> -f
   -
   name: <name.of.service>
   ports:
     - port_number: <port-number>
       protocol: tcp
   EOM
   ```

   <name.of.service> is the service name, for example **dashboard.myenterprise.acme.com**.
   <port-number> is the port number for the application, for example **8080**.

3. Get the service ID:

   ```
   palmctl get services --application-name <application-name>
   ```

4. Get the deployment ID:

```
palmctl get deployments --application-name <application-name>
```

5. Create the service endpoint:

```
cat << EOM | palmctl create service-endpoint --application-name <application-
name> \
--deployment-id <deployment-id> -f -
> local_service_ip_address: 127.0.0.1
> service_id: <service-id>
> EOM
```

For example:

```
cat << EOM | palmctl create service-endpoint --application-name my-backend-
app \
--deployment-id depl-95e74701-07aa-4d32-97c2-dfbf1eb5f6a7 -f -
> local_service_ip_address: 127.0.0.1
> service_id: svc-aa431961-c0e4-4ed6-875a-6435ebaef962
> EOM
```

Create a connection policy

The policy references the network segment where the gateway is deployed, and the service whose requests you want to route. Before you create the policy, get the IDs of the service and the network segment.

Complete the following steps:

1. Get the ID of the network segment:

   ```
   palmctl get networksegments
   ```

   The ID of the network segment is shown in the **`resource_id`** field in the output.

2. Get the ID of the service:

   1. First get the details of the application that contains the service:

      ```
      palmctl get applications
      ```

      The application name and **`resource_id`** are shown in the output.

   2. Get the ID of the service:

      ```
      palmctl get services --application-name <application-name>
      ```

      The ID of the service is shown in the **`resource_id`** field in the output.

3. Create the connection policy:

```
cat << EOM | palmctl create policy -f -
name: <policy-name>
action: Allow
network_segment_id: <network-segment-id>
from:
  type: networkSegment
  network_segment:
    network_segment_id: <network-segment-id>
to:
  type: service
  service:
    service_id: <service-id>
```

```
    application: <application-id>
EOM
```

<policy-name> must be unique.

CLI guide

Use the **palmctl** command-line interface (CLI) to configure and manage resources in your IBM Hybrid Cloud Mesh (Mesh) environment. This content gives an overview of using the Mesh CLI, with some examples.

Prerequisites

- Install and configure the CLI

Summary

The CLI carries out actions on resources, using the following command syntax:

```
palmctl <action> <resource> [<arguments>] [<flags>]
```

The CLI's context sensitive help shows a list of available commands and actions, and a summary of command usage. To view the **palmctl** help, use the **--help** or **-h** flag:

```
palmctl --help
```

To get help for a specific command, append it with the **--help** flag. For example:

```
palmctl config --help
```

The **palmctl** output format is YAML by default. The yq command is a useful open source package to parse **palmctl** output. If you prefer the JSON format, use the **-F json** flag to have **palmctl** output JSON format. In this case, the jq command is a useful open source package to parse **palmctl** JSON output.

You can see more detailed output using the **-v** or **--verbose** flag to indicate the verbosity level:

- 0: Error - displays only errors. This is the default.
- 1: Warning - also displays warning messages.
- 2: Info - also displays informational messages.
- 3: Debug - also displays debug messages.
- 4: Trace - also displays low level tracing of HTTP requests.

Example:

```
palmctl <action> <resource> --verbose 4
```

`palmctl` actions

create

Creates the specified resource and configures it according to a YAML input file.

get

Get the details of one or more resources.

`update`

Changes the configuration of the specified resource according to a YAML input file. This is a **full replacement** of the resource, meaning fields that are not specified in the input file will be set to empty.

`patch`

Updates the specified fields in the specified resource. Fields that are not updated will retain their current value.

`delete`

Deletes the specified resource. This command will not show any output unless verbosity is increased using the `-v <verbosity>` flag.

`bind`

Associates related resources. Use `bind` to associate permissions to roles, and roles to identities.

`unbind`

Decouples resources previously associated with each other using the `bind` command.

`redeploy`

Redeploys a resource, for example, a gateway.

`palmctl` resources

Use `palmctl <action> -h` to see the resources that can be used with that action.

`palmctl` usage examples

The following commands are examples of common uses of `palmctl`.

In the examples that require file input, this common pattern is used:

```
cat << EOM | palmctl <action> <resource> <arguments> <flags> -f -
name: some-name
description: some description
EOM
```

In this pattern the lines following the initial command (not including the final `EOM`) are sent to `palmctl` as an input file. The details follow:

- The `cat` command is instructed to read input from the following lines until it reaches `EOM`. It is the `<< EOM` argument that instructs it to do this.
- The `cat` command echoes the input it read to its output.
- The pipe command `|` sends the output of the `cat` command to the input of the `palmctl` command.
- The `-f -` flag instructs `palmctl` to read its input as if it was a file.

Creating an application resource

Register your application with Mesh:

```
cat << EOM | palmctl create application -f -
name: myapp
app_identity: app1.my.domain.palmetto.ibm.com
labels:                  # optional
```

```
  - app:my-app-label
EOM
```

Updating an application resource

Fully replace an application resource with new field values:

```
cat << EOM | palmctl create application -f -
name: myapp
app_identity: app1.different.domain.palmetto.ibm.com
labels:                   # optional
  - app:my-app-label
EOM
```

Updating one field in an application resource

```
cat << EOM | palmctl patch application --name myapp -f -
app_identity: app1.another.domain.palmetto.ibm.com
EOM
```

Get an application resource by name

```
palmctl get application --name myapp
```

Getting all application resources

```
palmctl get applications
```

Deleting an application resource by name

```
palmctl delete application --name myapp
```

Note: If the command is successful, there is not output.

Assigning (binding) a role to an identity (user)

Give a user all of the permissions contained in a role:

```
palmctl bind identity --name <user-email> --role-name ReadAll
```

Removing (unbinding) a role from an identity (user)

Remove a role's permissions from a user:

```
palmctl unbind identity --name <user-email> --role-name ReadAll
```

Installing and configuring the CLI

Download and install the latest version of the IBM Hybrid Cloud Mesh (Mesh) CLI. CLI packages for the following platforms are available:

- Red Hat® Enterprise Linux® (RHEL)
- Ubuntu
- Mac OSX
  - amd64
  - arm64
- Windows operating system

See Supported platforms for specific versions that are supported.

Prerequisites

- In order to use IBM® Hybrid Cloud Mesh, configure the IBM secrets manager. See Secrets manager.
- Create an API key. See Managing API keys.
- Set the API key in a variable for subsequent steps:

```
MESH_API_KEY=<your-api-key>
```

Downloading and installing the CLI

Downloading and installing the CLI on Ubuntu

```
PALMCTL_FILE_NAME=palmctl_latest_amd64.deb
curl -sSfLO
https://github.com/IBM/palmctl/releases/latest/download/$PALMCTL_FILE_NAME
sudo apt install "$PWD/$PALMCTL_FILE_NAME"
```

Downloading and installing the CLI on RHEL

```
PALMCTL_FILE_NAME=palmctl_latest_x86_64.rpm
curl -sSfLO
https://github.com/IBM/palmctl/releases/latest/download/$PALMCTL_FILE_NAME
sudo rpm -i "$PWD/$PALMCTL_FILE_NAME"
```

Downloading and installing on Mac OSX amd64

```
PALMCTL_FILE_NAME=palmctl_latest_macos_amd64.tar.gz
curl -sSfLO
https://github.com/IBM/palmctl/releases/latest/download/$PALMCTL_FILE_NAME
tar -xvf $PALMCTL_FILE_NAME
sudo ./palmctl/install.sh
source /usr/local/etc/bash_completion.d/bash_palmctl_completion
```

Downloading and installing on Mac OSX arm64

```
PALMCTL_FILE_NAME=palmctl_latest_macos_arm64.tar.gz
curl -sSfLO
https://github.com/IBM/palmctl/releases/latest/download/$PALMCTL_FILE_NAME
tar -xvf $PALMCTL_FILE_NAME
sudo ./palmctl/install.sh
source /usr/local/etc/bash_completion.d/bash_palmctl_completion
```

Mac OSX notes:

- To enable auto-completion in the `zsh` or `fish` shells, see the output of the `sudo ./palmctl/install.sh` command.
- If you download the `palmctl` package using the browser, when you run `palmctl`, OSX will give you the error message `"palmctl" cannot be opened because the developer cannot be verified`. To enable it, open `System Settings`, go to the `Privacy & Security` tab, and scroll to the `Security` section. You will see `"palmctl" was blocked from use because it is not from an identified developer`. Click `Allow Anyway`. (The `palmctl` OSX package will be notarized in a future release.)

Downloading and installing on Windows

Before you begin

The Windows installation script includes an optional step to install the completion feature for the Windows PowerShell console. If you use PowerShell, ensure that the PowerShell execution policy is set to a level that allows scripts to run. You can set the execution policy by running the following command in a PowerShell console:

```
Set-ExecutionPolicy RemoteSigned
```

Procedure

Complete the following steps to install the `palmctl` CLI:

1. Download the latest archive from this location.
2. Double-click the archive file to uncompress and extract the contents.
3. Open the command prompt and navigate to the `palmctl` folder.
4. Run `install.bat` to start the installation.
5. (Optional) If you use PowerShell, enter "Y" when you are prompted `Do you want to install powershell completion for palmctl[Y/N]?`

Uninstalling on Windows

Complete the following steps:

1. To uninstall `palmctl`, run `palmctl\uninstall.bat`.
2. If you installed the PowerShell completion feature, you must edit the PowerShell profile to remove the completion script. Use the following PowerShell command to edit the profile:

```
notepad $PROFILE
```

Verifying the CLI installation:

```
palmctl --version
```

Configuring the CLI

The `palmctl` CLI uses the following settings:

- User configuration: API key for user authentication
- Endpoint configuration: URL of Mesh
- Log configuration: Log setting management

The configuration is stored in `$HOME/palmctl_config.yaml`, by default. It is possible to change the CLI settings by:

- Using `palmctl` commands
- Exporting environment variables
- Changing the `PALMCTL_CONFIG_FILE` variable to point to a different configuration file
- Editing the config file directly

Changing the `palmctl` CLI settings using `palmctl` commands

Set your user API key:

```
palmctl config user --token $MESH_API_KEY
```

Set the endpoint URL:

```
palmctl config endpoint --url https://app.hybridcloudmesh.ibm.com
```

[Optional] Set the TLS client certificate pair:

```
palmctl config endpoint --client-cert <path/to/client.crt>
palmctl config endpoint --client-key <path/to/client.key>
```

[Optional] Set the log file (all output will be redirected to the log file):

```
palmctl config logs --log-file <path/to/file.log>
```

Changing the CLI settings using environment variables

Any environment variable automatically overrides the corresponding value in the config file.

These variables correspond with the listed actions:

- `PALMCTL_CONFIG_FILE`: Change the location of the configuration file
- `PALMCTL_USER_TOKEN`: Set the user API key
- `PALMCTL_ENDPOINT_URL`: Set the Mesh URL
- `PALMCTL_ENDPOINT_CA_CERT_FILE`: Set the CA cert file
- `PALMCTL_ENDPOINT_CLIENT_CERT_FILE`: Set the client cert file
- `PALMCTL_ENDPOINT_CLIENT_KEY_FILE`: Set the client key file
- `PALMCTL_LOG_FILE`: Set a log file to use.

For example, to change the location of the CLI configuration file, set the `PALMCTL_CONFIG_FILE` environment variable:

```
export PALMCTL_CONFIG_FILE=/another/path/to/config.yaml
```

Editing the configuration file directly

Edit the values directly in the YAML configuration file.

Example configuration file:

```
user:
  token: token_string
endpoint:
  url: {{site.data.keyword.console_url}}
  cacertfile: ""
  clientcertfile: ""
  clientkeyfile: ""
logs:
  file: ""
```

Managing API keys

To authenticate requests to the `palmctl` command-line interface, you must create a IBM Hybrid Cloud Mesh (Mesh) API key.

Creating API keys

Create an API key by completing the following steps:

1. Log in to the Mesh console.
2. Click **Manage** on the Mesh console home page and select **API Keys**.
3. Click **Create API key**.
4. Enter the API key information and click **Generate API key**.

For more information about installing and configuring `palmctl`, see Install the CLI.

Configuring API keys

Configure `palmctl` to use your API key by using the following command:

```
palmctl config user --token <api-key>
```

Viewing and deleting API keys

You can view and delete your API keys by using the API keys page in the Mesh console.

Managing secrets

A user with the tenant administrator role can configure secrets manager, and create internal and external secrets for the tenants in IBM Hybrid Cloud Mesh (Mesh).

Configure secrets manager

You can set the secrets manager type to `external` or `internal` by configuring the secrets manager. By default, the secrets manager type is set to `internal`. Complete the following steps to configure the secrets manager.

1. To get the current secrets manager configuration, run the following command:

   ```
   palmctl get secrets-manager-configuration
   ```

   If the type of your secrets manager configuration is `internal`, output similar to the following example is shown:

   ```
   sm_type_strategy: internal
   description: "<an internal configuration description>"
   ```

2. To retrieve the input parameters that are used to update the secrets manager configuration, run a command like this:

   ```
   cat secrets-manager-configuration.yaml
   ```

   For example, in the configuration file, if the type of your secrets manager configuration is set as `external`, you might see an output like this sample output:

   ```
   sm_type_strategy: "external"
   description: "<an external configuration description>"
   ```

```
api_key: "<sample-api-key>"
```

When you configure the external secrets manager, you must provide an API key.

3. To update the secrets manager configuration, run a command like this:

   ```
   palmctl update secrets-manager-configuration -f secrets-manager-
   configuration.yaml
   ```

   When your secrets manager configuration is updated to **external**, output similar to the following example is shown:

   ```
   sm_type_strategy: "external"
   description: "<an external configuration description>"
   ```

4. When the secrets of type external is already stored in Mesh, and the type of the secrets manager configuration is changed to internal, you can choose to edit the API key. You can provide a new API key that overrides the last used API key. In case a new API key is not provided, last used API key is used.

   Run a command like this:

   ```
   palmctl update secrets-manager-configuration -f
   secrets-manager-configuration-api-key.yaml
   ```

   Output similar to the following example is shown:

   ```
   sm_type_strategy: internal
   description: "<internal secret configuration description>"
   api_key: "<updated_api_key_for_existing_secrets>"
   ```

Create secrets

You can create secrets based on the type of secrets manager that you configure for your tenants. When you change from one secrets manager type to another, you must modify the configuration of the secrets manager.

For more information about how to create secrets, see the following topics:

- Creating external secrets
- Creating internal secrets

Creating external secrets

When you create external secrets, you must provide an API key and specify the type of secrets manager as **external** in the secrets manager configuration.

Procedure

Complete the following steps to create an external secret for a tenant:

1. To retrieve the input parameters that can be used to create an external secret, run a command like this:

   ```
   cat <input_file.yaml>
   ```

   <input_file.yaml> is the YAML file that contains input parameters.

For example, run a command like this to retrieve the input parameters from the `test_secret_ext.yaml` file:

```
cat test_secret_ext.yaml
```

Output similar to the following example is shown:

```
name: my-ibm-secret-external
type: cloud-ibm
path: https://instance-id.eu-de.secrets-
manager.appdomain.cloud/api/v2/secrets/kv/215b65c1-381c-0b88-af5c-
560551207f5e
resource_group_id: default-infra
```

2. To create an external secret, run the following command:

   ```
   palmctl create secret -f <input_file.yaml>
   ```

   For example, run a command like this to create an external secret from the input parameters file `test_secret_ext.yaml`:

   ```
   palmctl create secret -f test_secret_ext.yaml
   ```

   Output similar to the following example is shown:

   ```
   ...
   name: my-ibm-secret-external
   path: https://instance-id.eu-de.secrets-
   manager.appdomain.cloud/api/v2/secrets/kv/215b65c1-381c-0b88-af5c-
   560551207f5e
   resource_group_id: default-infra
   type: cloud-ibm
   ...
   ```

Creating internal secrets

When you create internal secrets, you must specify the type of secrets manager as **`internal`** in the secrets manager configuration.

Procedure

Complete the following steps to create an internal secret for a tenant:

1. To retrieve the input parameters that can be used to create an internal secret, run a command like this:

   ```
   cat <input_file.yaml>
   ```

   <input_file.yaml> is the YAML file that contains input parameters.

   For example, run a command like this to retrieve the input parameters from the `test_secret_int.yaml` file:

   ```
   cat test_secret_int.yaml
   ```

   Output similar to the following example is shown:

```
name: my-ibm-secret-internal
secret:
 apikey: "<sample-api-key>"
resource_group_id: default-infra
type: cloud-ibm
```

2. To create an internal secret, run the following command:

   ```
   palmctl create secret -f <input_file.yaml>
   ```

   For example, run a command like this to create an internal secret from the input parameters file `test_secret_int.yaml`:

   ```
   palmctl create secret -f test_secret_int.yaml
   ```

   Output similar to the following example is shown:

   ```
   ...
   name: my-ibm-secret-internal
   resource_group_id: default-infra
   secret_manager_type: internal
   type: cloud-ibm
   ...
   ```

Creating Kubernetes secret

Prerequisites

- Ensure Your Kubernetes/OCP Cluster is up in your on-prem and Kubectl command is running.
- Ensure you have access to your VMWARE/KVM to orchestrate a VM using the VMWARE based Gateway Image.

Procedure

Complete the following steps to create a Kubernets secret for your cluster:

1. Prepare application cluster credentials.

   - Get the `kubeconfig file` on Cluster VM.

   ```
   cd
   kubectl config view --minify --raw > <kube-config-file.yaml>

   Example:
   kubectl config view --minify --raw > xyz.yaml
   ```

2. Extract secrets as secretData.json file.

   Copy the kube-config-file.yaml from the first step locally to your host, where `palmctl` is installed and run the this command to extract Kubernetes secret:

   ```
   palmctl check kubeconfig -f kube-config-file.yaml --extract

   Example:
   % palmctl check kubeconfig -f xyz.yaml --extract
   api_end_point: https:0.0.0: xyz
   ```

```
secrets:
- k8sClientToken
- k8sCA

% cat secretData.json | jq
{
  "k8sCA": "xxxxxxxxxxxxxxx",
  "k8sClientToken": "xxxxxxxxxxxxx"
}
```

3. Create a Kubernetes-type secret in IBM Cloud® if your secrets are stored in external IBM® secret manager.

   Complete the following steps to access the IBM Secret Manager and create Kubernetes-type secret:

   1. Go to IBM Cloud Console.
   2. Click on the **Resource list**.
   3. Scroll down and expand **Security** option.
   4. Select the secret manager from the displyed list to view secrets.
   5. Click **Add +**.
   6. Select **Key-value** option and click **Next**.
   7. Provide a name and click **Next**
   8. Click on **select file** tab and upload the Key-value data from `secretData.json` generated in previous step.
4. Add Kubernetes secret in Mesh console.

   Complete the following steps to add the Kubernetes secret:

   1. On the Mesh console go to **Admin**.
   2. Select **Secrets** from the Admin dropdown.
   3. On the **Secrets** page click **Register Secret**.
   4. Provide a secret Name, Choose the secret type as Kubernetes.
   5. Provide the path to secret as explained in the next steps.

   To get the secret path, follow these steps:

   1. In the IBM Secrets Manager console, click the secret you created.
   2. In the **Details** side-panel, click the **Actions** menu, and click **Show snippet**.
   3. Click **Curl** tab.
   4. Copy the URL in the curl command.

   Provide the copied path to the Mesh console, when registering a secret to the external secret manager.

   If you are using the internal secret manager, provide the value of k8sCA, k8sClientToken, and other parameters from secretData.json extracted in Step 2 directly in the Mesh console as explained in the next steps:

   1. On the Mesh console go to **Admin**.
   2. Select **Secrets** from the **Admin** dropdown.
   3. On the **Secrets** page, click **Register Secret**. If you are using the internal secret manager, Hybrid Cloud Mesh Secrets Manager will be displayed in **Stored in** field.
   4. Provide a secret name and choose the secret type as Kubernetes.
   5. Provide the value of parameters such as K8Ca (Optiona), k8sClientToken, K8sClientKey and K8sClientCert extracted in Step 2.

   Either k8sClientToken or (K8sClientKey and K8sClientCert) pair is mandatory to register secret using the internal secret manager using Mesh console.

Registering infrastructure resources

In order for IBM Hybrid Cloud Mesh (Mesh) to connect your applications, it requires knowledge about the cloud and on-prem infrastructure where the applications are running.

There are two ways to inform Mesh about your infrastructure:

- Provide a limited read-only set of credentials to your infrastructure so Mesh can auto-discover it. This procedure is described in Discovering cloud infrastructure.
- Manually register the relevant parts (resources) of your infrastructure with Mesh. This procedure is described here.

Prerequisites

- Install and configure the CLI

Overview

Your infrastructure resources have a hierarchy, and it is necessary to register the resources in order, from the broadest to the most specific:

- cloud
    - location
    - vpc (virtual private cloud)
        - securitygroup
    - cluster
        - namespace
    - node (virtual machine or bare metal)

Deployment environment is the abstract term Mesh uses for Virtual Private Cloud (VPCs), clusters, and nodes. The abstract term for security groups and namespaces is partition. The primary purpose of registering your infrastructure is to describe the deployment environments and partitions where your applications run.

Registering clouds

Register each cloud that contains applications you want Mesh to connect.

Example:

```
cat << EOM | palmctl create cloud -f -
name: AWS
type: AWS
is_private: false
EOM
```

Notes about the cloud fields:

- The value of the `name` field can be anything, as long as it is unique.
- The valid values for `type` are: `IBM`, `AWS`, `other`.
- To represent all of your on-prem data centers:
    - Set `type` to `other`
    - Set `is_private` to `true`

Registering cloud locations

Register each location within each cloud that contains your applications. Registering a location is easier if it is one of the known public cloud locations. First, list the public cloud locations by running **`palmctl get cloudlocations`** and note the **`code`** field of the cloud location that corresponds to your location. Then, use that code in the **`cloud_location_code`** field when registering your location.

Example of registering a location associated with an existing public cloud location:

```
cat << EOM | palmctl create location --cloud-name AWS -f -
name: AWS-east
cloud_location_code: abcdef
is_multi_zone: true
EOM
```

When registering a location that is not associated with an existing public cloud location (usually an on-prem data center), specify additional fields.

Example of registering a location not associated with an existing public cloud location:

```
cat << EOM | palmctl create location --cloud-name OnPrem -f -
name: Factory-CA
type: region
is_multi_zone: false
city: Los Angeles
administrative_region: California
country: United States
geo_coordinates: 34.1139,-118.4068
EOM
```

Note: The valid values for the **`type`** field are **`site`**, **`region`**, **`zone`**.

Registering VPCs

Register each VPC that contains your applications. A VPC can be a deployment environment itself (for example, has applications on virtual machines (VMs) in the VPC) or just a container for Kubernetes clusters. In both cases, it should be registered. Use the **`infra_only`** field to distinguish between the two cases.

Example:

```
cat << EOM | palmctl create vpc --cloud-name AWS -f -
name: myvpc
location_id: <resource_id from previous command>
type: AWSVPC
api_end_point: https://mycloud.com/vpc1
infra_only: false
EOM
```

Registering VPC security groups

If a VPC is partitioned into security groups, register each security group that contains your applications.

Example:

```
cat << EOM | palmctl create securitygroup --cloud-name AWS --vpc-name 'myvpc' -f -
name: security group 1
EOM
```

Registering Kubernetes clusters

Register each cluster that is running your applications. If **api_end_point**, **credentials_key**, and **auto_discover: true** are specified, then Mesh will automatically discover the namespaces and applications running in the cluster.

The name of the cluster that you create in Mesh must match with the name of the cluster that you deployed in IBM Cloud® or AWS.

Example:

```
cat << EOM | palmctl create cluster --cloud-name AWS -f -
name: myEKS
location_id: <resource_id from previous command>
type: EKS
is_multi_zone: true
api_end_point: https://mycloud.com/cluster1
credentials_key: /path/to/key/in/SM
auto_discover: true
EOM
```

Registering cluster namespaces

If you did not configure auto-discovery when registering the clusters, register each namespace in them that is running your applications.

Example:

```
cat << EOM | palmctl create namespace --cloud-name AWS --cluster-name myEKS -f -
name: namespace 1
EOM
```

Registering nodes

For VMs that are running your applications and are outside of a VPC, register a node resource for it.

Example:

```
cat << EOM | palmctl create node --cloud-name AWS -f -
name: myVM
location_id: <resource_id from previous command>
type: VM
EOM
```

What's Next

- Use the following commands to get help viewing, modifying, or deleting the infrastructure resources you have created:
  - `palmctl get -h`
  - `palmctl update -h`
  - `palmctl delete -h`
- Go to Registering application resources to make Mesh aware of the applications it should provide connectivity for.

Registering application resources

In order for IBM Hybrid Cloud Mesh (Mesh) to connect your applications, it requires knowledge about where those applications are running.

There are two ways to inform Mesh about your applications:

1. Provide a limited read-only set of credentials to each deployment environment (VPC, cluster, or node) so Mesh can auto-discover the applications running on them. This procedure is described in Discovering cluster namespaces and application.
2. Manually register your applications with Mesh. This procedure is described here.

Prerequisites

- Install and configure the CLI

Overview of registering applications

Your applications and their deployments have a hierarchy, and it is necessary to register the application resources in order, from the broadest to the most specific:

- application
    - service
    - deployment
    - instance
    - service-endpoint

Registering applications

Register each application you want Mesh to connect. An application can have multiple services, deployments, instances, and service endpoints. Those will be registered in subsequent steps.

Example:

```
cat << EOM | palmctl create application -f -
name: myapp
app_identity: app1.my.domain.palmetto.ibm.com
EOM
```

Note: The `app_identity` field is a fully qualified domain name (FQDN) that uniquely identifies the application.

Registering services

Register each service that an application provides.

Example:

```
cat << EOM | palmctl create service --application-name myapp -f -
name: myservice.my.domain.palmetto.ibm.com
ports:
  - port_number: '9050'
```

```
    protocol: TCP
EOM
```

Note: The `name` field must be the FQDN of the service, which must be modified to end with palmetto.ibm.com. This may require a new TLS certificate depending on the Subject Alternative Names used.

Registering deployments

Register where each application (including its services) is deployed.

Example:

```
cat << EOM | palmctl create deployment --application-name myapp -f -
depl_env_id: <deployment-environment-resource-id>
partition_id: <partition-resource-id>
EOM
```

Notes about the deployment fields:

- Run the `palmctl get deploymentenvs` command to find the deployment environment's `resource_id` value and substitute it for `<deployment-environment-resource-id>`.
- Run the `palmctl get partitions --deployment-env-id <deployment-environment-resource-id> --cloud-name cloud1` command to find the partition's `resource_id` value and substitute it for `<partition-resource-id>`.

When the command completes, note the `resource_id` of the created deployment and substitute it for `<deployment-resource-id>` in the following steps.

Registering instances

Register each of the application instances in each deployment. In Kubernetes clusters, the instances are the pods for application. In VPCs, the instances are the VMs that are running the application.

Example:

```
cat << EOM | palmctl create instance --application-name myapp --deployment-id
<deployment-resource-id> -f -
ip_address: 10.4.5.6
EOM
```

Note: `ip_address` is the local IP address of this instance within the deployment environment.

Registering service endpoints

Register each of the endpoints a service listens on.

Example:

```
cat << EOM | palmctl create service-endpoint --application-name myapp --
deployment-id <deployment-resource-id> -f -
service_id: <service-resource-id>
local_service_ip_address: 10.7.8.9
EOM
```

Notes about the service-endpoint fields:

- Run a command like `palmctl get service --name myservice.my.domain.palmetto.ibm.com --application-name myapp` to find the service's `resource_id` value and substitute it for `<service-resource-id>`.
- `local_service_ip_address` is the local IP address within the deployment environment that is being proxied to each of the instances.

What's Next

- Use the following commands to get help viewing, modifying, or deleting the application resources you have created:
  - `palmctl get -h`
  - `palmctl update -h`
  - `palmctl delete -h`
- Go to Managing applications connections to have Mesh connect your applications.

Frequently asked questions

The following are answers to some frequently asked questions (FAQs) about IBM Hybrid Cloud Mesh (Mesh).

- Which are the deployment environments that Mesh edge gateways support?
- Which firewall ports are opened for an on-prem Mesh edge gateway deployment?
- Which are the deployment environments that Mesh waypoint gateways support?
- Can I deploy Mesh gateways in IBM Cloud® or AWS without providing the cloud secrets for autodiscovery or autodeploy?
- Why do I need to provide my cloud credentials?

Which are the deployment environments that the Mesh edge gateways support?

You can deploy the Mesh edge gateways in the following deployment environments:

- AWS
  - Elastic Kubernetes Service (EKS)
- IBM Cloud
  - IBM Cloud Kubernetes Service (IKS)
  - IBM Cloud Red Hat® OpenShift® Kubernetes Service (ROKS)
- VMWare vSphere (on-prem)
  - Red Hat OpenShift Container Platform (OCP)

VMWare vSphere OCP supports only Calico Container Network Interface (CNI).

For more information about gateways, see Working with gateways.

Which firewall ports are opened for an on-prem Mesh edge gateway deployment?

The following firewall ports are opened for the ingress traffic for an on-prem Mesh edge gateway deployment:

Proto	Port	Source	Purpose	Comment
TCP	7777	0.0.0.0/0	SSH	GW's internal firewall restricts access to selected hosts
UDP	4500	0.0.0.0/0	IPsec and IKEv2	Traffic coming from other GWs on secure overlay
UDP	6081	0.0.0.0/0	GENEVE tunnels	Traffic coming from other GWs on regular overlay

For more information, see Gateway network requirements for IBM® Hybrid Cloud Mesh.

Which are the deployment environments that the Mesh waypoint gateways support?

You can automatically deploy the Mesh waypoint gateways in the following deployment environments:

- AWS
  - Elastic Kubernetes Service (EKS)
- IBM Cloud
  - IBM Cloud Kubernetes Service (IKS)
  - IBM Cloud Red Hat OpenShift Kubernetes Service (ROKS)

You can manually deploy the Mesh waypoint gateways in the following deployment environment:

- VMWare vSphere (on-prem)
  - Red Hat OpenShift Container Platform (OCP)

Can I deploy the Mesh gateways in IBM Cloud or AWS without providing the cloud secrets for autodiscovery or autodeploy?

No you can’t deploy the Mesh gateways in IBM Cloud or AWS without providing the cloud secrets. To deploy the Mesh gateways in IBM Cloud or AWS, you must provide the cloud credentials with appropriate permissions.

Why do I need to provide my cloud credentials?

Mesh needs to interact with your cloud environments, and needs permission to create gateways on your deployment environments. It also needs access to discover your cloud infrastructure and applications.

Support information

If you have a problem with your IBM® software, you want to resolve it quickly. IBM provides multiple ways for you to obtain the support you need.

Online

Go to the IBM Software Support site at Products Help and follow the instructions.

IBM Support Assistant

The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. For more information about how to open a support case, see Opening a support case with IBM Support.

Opening a support case with IBM® Support

If you cannot find a solution to your problem in product documentation, open a support case with IBM Support.

Before you open a support case

Before you contact IBM Support, review the Release notes to check whether your issue is already reported and a workaround or solution exists in the documentation.

Opening a support case

To open a support case, complete the following steps:

1. Go to the IBM Support site and click **Open a case**.
2. Log in with your IBM ID and password, and consent to the IBM Privacy Statement.
3. Enter the following case details:
    1. Case title that summarizes your issue.
    2. Product information.
    3. Severity of the issue and its business impact.
    4. Detailed description of your issue in your preferred language that include product version number, problem, steps to reproduce the issue, suggestions, and expected outcome.
    5. Your contact details.

   The case severity is based on the business impact of the problem. If you set the case severity as 1, you need to be available 24x7 to work with IBM Support on this issue.
4. Optional: To save this case as a template for future reference, select the **Case template** checkbox.
5. Click **Submit a case**.

Glossary

This glossary provides terms and definitions for IBM Hybrid Cloud Mesh.

The following cross-references are used in this glossary:

- *See* refers you from a non-preferred term to the preferred term or from an abbreviation to the spelled-out form.

- *See also* refers you to a related or contrasting term.

A C D E G I K M N O P R S U W

A

API key

A unique code that is passed to an API to identify the calling application or user. An API key is used to track and control how the API is being used, for example, to prevent malicious use or abuse of the API.

application

One or more computer programs or software components that provide a function in direct support of a specific business process or processes.

application deployment

The instantiation of an application within a deployment environment.

application group

A set of application resources, such as applications including their services, deployments, or policies, that are grouped to simplify the process of providing access permissions. A user who has access to an application group automatically has access to all resources in the group.

application resource group

See application group.

autodiscover

To automatically detect and register resources.

availability zone

- An isolated data center that is located within a multizone region in which public cloud services originate and operate.
- An operator-assigned, functionally independent segment of network infrastructure.

C

connection

In data communication, an association established between entities for conveying information.

connection access policy

A collection of one or more rules that run at an edge gateway to determine which applications are allowed to talk to which services. These rules might influence the network path taken and whether or not the path is encrypted.

D

deployment

A model or application package that is available for use.

deployment environment

A cluster, VPC or computing environment where applications can be hosted.

DevOps

A software methodology that integrates application development and IT operations so that teams can deliver code faster to production and iterate continuously based on market feedback.

$

discovery secret name

A secret that the autodiscovery process uses to access protected resources.

E

edge gateway

A network service that runs at the edge of the network and provides network functions such as network termination, network address translation, firewall policy enforcement, and tunneling.

event

An occurrence of significance to a task or system. Events can include completion or failure of an operation, a user action, or the change in state of a process.

external network fabric

The underlay networking used to transport traffic between gateways.

G

gateway

A device or program used to connect networks or systems with different network architectures.

Grafana

An open source analytics and visualization platform to monitor, search, analyze, and visualize metrics.

I

IBM Cloud®

An open-standards, cloud-based platform for building, managing, and running apps of all types, such as web, mobile, big data, and smart devices. Capabilities include Java™, mobile back-end development, and application monitoring, as well as features from ecosystem partners and open source—all provided as-a-service in the cloud.

infrastructure group

A set of infrastructure resources, such as clouds, deployment environments, or gateways, that are grouped to simplify the process of providing access permissions. A user who has access to an infrastructure group automatically has access to all resources in the group.

infrastructure resource group

See infrastructure group.

instance

A runtime occurrence of an application deployment. An application can have multiple instances of an application deployment running in a deployment environment.

K

K8s

See Kubernetes.

Kubernetes (K8s)

An open-source orchestration tool for containers.

M

managed

Referring to infrastructure resources such as deployment environments and partitions that are visible by default on all lists that are used in the management of the IBM® Hybrid Cloud Mesh network.

multicluster service

An enterprise service that has service instances within one or more clusters, and is accessed from consumer application components that are not co-resident in the same cluster as the provided service Instance.

N

NetOps

A networking approach that incorporates DevOps techniques such as automation, virtualization, and orchestration to improve network agility.

network fabric

A network topology in which components pass data to each other through interconnecting switches.

O

organization

A business entity with a specific role or roles that interact with other organizations in a supply chain to conduct business. An organization represents any unit of a business whether it is a company, legal entity, a business group, sales organization, purchasing organization, or warehouse.

overlay networks

A software-defined network that uses network virtualization to build connectivity on top of the physical network using tunneling encapsulations.

P

partition

A mechanism within a deployment environment that isolates workloads from one another using policies to restrict access to partitions or between partitions.

policy

A set of considerations that influence the behavior of a managed resource or a user.

R

repo

See repository.

repository (repo)

A persistent storage area for data and other application resources.

resource

A physical or logical component that can be provisioned or reserved for an application or service instance. Examples of resources can include storage, processors, memory, clusters, and VMs.

S

SaaSops

A set of processes and skills to manage and secure a SaaS environment through centralized and automated operations.

secret

A type of sensitive information, such as a password or an API key, that is used by an application to access a protected resource.

service

- A set of capabilities that provide reusable functions.
- A process or function that is provided by an application on a specific network port.

service endpoint

The physical address of a service that implements one or more interfaces.

subject

A consumer or provider of services, such as an application, user, device, or system.

U

underlay network

An underlying physical network responsible for the delivery of packets across networks.

W

waypoint gateway

A Mesh gateway that directs network traffic through specific routes through the use of strategic Internet Gateway Protocol values to improve network performance or save network cost. Together with the other Mesh gateways, waypoint gateways form an overlay network for applications.

Notices

These legal notices pertain to the IBM Hybrid Cloud Mesh product and its documentation.

This information was developed for products and services offered in the US. This material might be available from IBM® in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBMproducts. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at Copyright and trademark information.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

VMware, the VMware logo, VMware Cloud Foundation, VMware Cloud Foundation Service, VMware vCenter Server, and VMware vSphere are registered trademarks or trademarks of VMware, Inc. or its subsidiaries in the United States and/or other jurisdictions.

Terms and conditions

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside of your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING

BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Licensing

For full details, see the License Information documents.