Multicluster

Deploy Gloo Mesh Core across multiple clusters to gain valuable insights into your Istio service meshes.

Gloo Mesh Core deploys alongside your Istio installations in single or multicluster environments, and gives you instant insights into your Istio environment through a custom dashboard.

You can follow this guide to quickly get started with Gloo Mesh Core. To learn more about the benefits and architecture, see About. To customize your installation with Helm instead, see the advanced installation guide.

Before you begin

1 Install the following command-line (CLI) tools.

`kubectl`, the Kubernetes command line tool. Download the `kubectl` version that is within one minor version of the Kubernetes clusters you plan to use.

`meshctl`, the Solo command line tool.

```
curl -sL https://run.solo.io/meshctl/install | GLOO_MESH_VE
export PATH=$HOME/.gloo-mesh/bin:$PATH
```

2 Create or use at least two existing Kubernetes clusters. The instructions in this guide assume one management cluster and two workload clusters.

The cluster name must be alphanumeric with no special characters except a hyphen (-), lowercase, and begin with a letter (not a number).

3 Set the names of your clusters from your infrastructure provider. If your clusters have different names, specify those names instead.

```
export MGMT_CLUSTER=mgmt
export REMOTE_CLUSTER1=cluster1
export REMOTE_CLUSTER1=cluster1
export REMOTE_CLUSTER2=cluster2
```

4 Save the kubeconfig contexts for your clusters. Run `kubectl config get-contexts`, look for your cluster in the `CLUSTER` column, and get the context name in the `NAME` column. **Note**: Do not use context names with underscores. The generated certificate that connects workload clusters to the management cluster uses the context name as a SAN specification, and underscores in SAN are not FQDN compliant. You can rename a context by running `kubectl config rename-context "<oldcontext>" <newcontext>`.

```
export MGMT_CONTEXT=<management-cluster-context>
export REMOTE_CONTEXT1=<remote-cluster1-context>
export REMOTE_CONTEXT2=<remote-cluster2-context>
```

5 Set your Gloo Mesh Core license key as an environment variable. If you do not have one, contact an account representative. If you prefer to specify license keys in a secret instead, see Licensing. To check your license's validity, you can run `meshctl license check --key $(echo ${GLOO_MESH_CORE_LICENSE_KEY} | base64 -w0)`.

```
export GLOO_MESH_CORE_LICENSE_KEY=<license_key>
```

Install Gloo Mesh Core

In a multicluster setup, you deploy the Gloo management plane into a dedicated management cluster, and the Gloo data plane into one or more workload clusters that run Istio service meshes.

Management plane

Deploy the Gloo management plane into a dedicated management cluster.

1 Install Gloo Mesh Core in your management cluster. This command uses a basic profile to create a `gloo-mesh` namespace and install the Gloo management plane components, such as the management server and Prometheus server, in your management cluster. For more information, check out the CLI install profiles.

```
meshctl install --profiles gloo-core-mgmt \
--kubecontext $MGMT_CONTEXT \
--set common.cluster=$MGMT_CLUSTER \
--set licensing.glooMeshCoreLicenseKey=$GLOO_MESH_CORE_LICENSE
```

> This guide assumes one dedicated management cluster, and two Istio workload clusters that you register with the management cluster. If you plan to register the management cluster so that it can also function as a workload cluster, include `--set telemetryGateway.enabled=true` in this command.

2 Verify that the management plane pods have a status of `Running`.

```
kubectl get pods -n gloo-mesh --context $MGMT_CONTEXT
```

Example output:

```
NAME                                         READY   STATUS    RE
gloo-mesh-mgmt-server-56c495796b-cx687       1/1     Running   0
gloo-mesh-redis-8455d49c86-f8qhw             1/1     Running   0
gloo-mesh-ui-65b6b6df5f-bf4vp                3/3     Running   0
gloo-telemetry-collector-agent-7rzfb         1/1     Running   0
gloo-telemetry-gateway-6547f479d5-r4zm6      1/1     Running   0
prometheus-server-57cd8c74d4-2bc7f           2/2     Running   0
```

3 Save the external address and port that your cloud provider assigned to the Gloo OpenTelemetry (OTel) gateway service. The OTel collector agents in each workload cluster send metrics to this address.

```
export TELEMETRY_GATEWAY_IP=$(kubectl get svc -n gloo-mesh glo
export TELEMETRY_GATEWAY_PORT=$(kubectl get svc -n gloo-mesh g
export TELEMETRY_GATEWAY_ADDRESS=${TELEMETRY_GATEWAY_IP}:${TEL
echo $TELEMETRY_GATEWAY_ADDRESS
```

Data plane

Register each workload cluster with the Gloo management plane by deploying Gloo data plane components. A deployment named `gloo-mesh-agent` runs the Gloo agent in each workload cluster.

1 Register both workload clusters with the management server. These commands use a basic profile to create a `gloo-mesh` namespace and install the Gloo data plane components, such as the Gloo agent. For more information, check out the **CLI install profiles**.

```
meshctl cluster register $REMOTE_CLUSTER1 \
  --kubecontext $MGMT_CONTEXT \
  --profiles gloo-core-agent \
  --remote-context $REMOTE_CONTEXT1 \
  --telemetry-server-address $TELEMETRY_GATEWAY_ADDRESS

meshctl cluster register $REMOTE_CLUSTER2 \
  --kubecontext $MGMT_CONTEXT \
  --profiles gloo-core-agent \
  --remote-context $REMOTE_CONTEXT2 \
  --telemetry-server-address $TELEMETRY_GATEWAY_ADDRESS
```

2 Verify that the Gloo data plane components in each workload cluster are healthy. If not, try **debugging the agent**.

```
meshctl check --kubecontext $REMOTE_CONTEXT1
meshctl check --kubecontext $REMOTE_CONTEXT2
```

Example output:

```
🟢 Gloo deployment status

Namespace | Name                           | Ready | Status
gloo-mesh | gloo-mesh-agent                | 1/1   | Healthy
gloo-mesh | gloo-telemetry-collector-agent | 3/3   | Healthy
```

3 Verify that your Gloo Mesh Core setup is correctly installed. If not, try **debugging the relay connection**. Note that this check might take a few seconds to verify that:

Your Gloo product licenses are valid and current.

The Gloo CRDs are installed at the correct version.

The management plane pods in the management cluster are running and healthy.

The agents in the workload clusters are successfully identified by the management server.

```
meshctl check --kubecontext $MGMT_CONTEXT
```

Example output:

```
🟢 License status

INFO  gloo-mesh-core enterprise license expiration is 25 Aug 2

🟢 CRD version check

🟢 Gloo deployment status

Namespace | Name                            | Ready | Status
gloo-mesh | gloo-mesh-mgmt-server           | 1/1   | Healthy
gloo-mesh | gloo-mesh-redis                 | 1/1   | Healthy
gloo-mesh | gloo-mesh-ui                    | 1/1   | Healthy
gloo-mesh | gloo-telemetry-collector-agent | 3/3   | Healthy
gloo-mesh | gloo-telemetry-gateway          | 1/1   | Healthy
gloo-mesh | prometheus-server               | 1/1   | Healthy

🟢 Mgmt server connectivity to workload agents

Cluster  | Registered | Connected Pod
cluster1 | true       | gloo-mesh/gloo-mesh-mgmt-server-65bd55
cluster2 | true       | gloo-mesh/gloo-mesh-mgmt-server-65bd55

Connected Pod                                 | Clusters
gloo-mesh/gloo-mesh-mgmt-server-65bd557b95-v8qq6 | 2
```

Deploy Istio

Check whether Istio control planes already exist in the workload clusters.

```
kubectl get pods -n istio-system --context ${REMOTE_CONTEXT1}
kubectl get pods -n istio-system --context ${REMOTE_CONTEXT2}
```

If `istiod` pods exist in each workload cluster, such as in this example output, you already installed Istio control planes. Continue to the next step.

```
NAME                        READY   STATUS    RESTARTS   AGE
istiod-b65676555-g2vmr      1/1     Running   0          8d
NAME                        READY   STATUS    RESTARTS   AGE
istiod-7b96cb895-4nzv9      1/1     Running   0          8d
```

If no `istiod` pod exists, you can use the Solo distribution of Istio to install a sidecar service mesh in each workload cluster. For more information, check out Solo distributions of Istio. For more information about service mesh lifecycle management with Gloo, check out Service mesh lifecycle.

1. Download the `gs-ilm-glm.yaml` example file, which contains basic `IstioLifecycleManager` configuration for the `istiod` control plane and `GatewayLifecycleManager` configuration for an Istio ingress gateway. For more information about the custom resources, see the API reference.

   ```
   curl -0L https://raw.githubusercontent.com/solo-io/gloo-mesh-u
   ```

2. Update the example file with the environment variables that you previously set, and save the updated file as `gs-ilm-glm-values.yaml`. For example, you can run a terminal command to substitute values:

   ```
   envsubst < gs-ilm-glm.yaml > gs-ilm-glm-values.yaml
   ```

3. Apply the `IstioLifecycleManager` and `GatewayLifecycleManager` CRs to your management cluster.

   ```
   kubectl apply -f gs-ilm-glm-values.yaml --context $MGMT_CONTEX
   ```

4. In each workload cluster, verify that the Istio pods have a status of `Running`.

   ```
   kubectl get pods -n istio-system --context $REMOTE_CONTEXT1
   kubectl get pods -n istio-system --context $REMOTE_CONTEXT2
   ```

Example output:

```
NAME                              READY   STATUS    RESTARTS   A
istiod-1-23-b65676555-g2vmr       1/1     Running   0          4
NAME                              READY   STATUS    RESTARTS   A
istiod-1-23-7b96cb895-4nzv9       1/1     Running   0          4
```

5 In each workload cluster, verify that the ingress gateway pods have a status of `RUNNING` and that the load balancer services have external addresses.

```
kubectl get pods,svc -n gloo-mesh-gateways --context ${REMOTE_
kubectl get pods,svc -n gloo-mesh-gateways --context ${REMOTE_
```

Example output for one cluster:

```
NAME                                      READY   STATUS    REST
istio-ingressgateway-665d46686f-nhh52     1/1     Running   0

NAME                          TYPE           CLUSTER-IP      EXT
istio-ingressgateway          LoadBalancer   10.96.252.49    <ex
```

Deploy a sample app

To analyze your service mesh with Gloo Mesh Core, be sure to include your services in the mesh.

If you already deployed apps that you want to include in the mesh, you can run the following command to label the service namespaces for Istio sidecar injection.

```
kubectl label ns <namespace> istio-injection=enabled --context
```

If you don't have any apps yet, you can deploy Bookinfo, the Istio sample app.

1 Create the `bookinfo` namespace in each cluster, and label the workload cluster namespaces for Istio injection so that the services become part of the service mesh.

```
kubectl create ns bookinfo --context ${REMOTE_CONTEXT1}
kubectl label ns bookinfo istio-injection=enabled --context
kubectl create ns bookinfo --context ${REMOTE_CONTEXT2}
kubectl label ns bookinfo istio-injection=enabled --context
```

2 Deploy Bookinfo with the `details`, `productpage`, `ratings`, `reviews-v1`, and `reviews-v2` services in `cluster1`.

```
# deploy bookinfo application components for all versions l
kubectl -n bookinfo apply -f https://raw.githubusercontent.
# deploy an updated product page with extra container utili
kubectl -n bookinfo apply -f https://raw.githubusercontent.
# deploy all bookinfo service accounts
kubectl -n bookinfo apply -f https://raw.githubusercontent.
```

3 Deploy Bookinfo with the `ratings` and `reviews-v3` services in `cluster2`.

```
# deploy reviews and ratings services
kubectl -n bookinfo apply -f https://raw.githubusercontent.
# deploy reviews-v3
kubectl -n bookinfo apply -f https://raw.githubusercontent.
# deploy ratings
kubectl -n bookinfo apply -f https://raw.githubusercontent.
# deploy reviews and ratings service accounts
kubectl -n bookinfo apply -f https://raw.githubusercontent.
```

4 Verify that the Bookinfo app deployed successfully.

```
kubectl get pods,svc -n bookinfo --context ${REMOTE_CONTEXT
kubectl get pods,svc -n bookinfo --context ${REMOTE_CONTEXT
```

Explore the UI

Use the Gloo UI to evaluate the health and efficiency of your service mesh. You can review the analysis and insights for your service mesh, such as recommendations to harden your Istio environment and steps to implement them in your environment.

Launch the dashboard

1. Open the Gloo UI. The Gloo UI is served from the `gloo-mesh-ui` service on port 8090. You can connect by using the `meshctl` or `kubectl` CLIs.

   meshctl kubectl

   For more information, see the CLI documentation.

   ```
   meshctl dashboard --context $MGMT_CONTEXT
   ```

2. Review your **Dashboard** for an at-a-glance overview of your Gloo Mesh Core environment. Environment insights, health, status, inventories, security, and more are summarized in the following cards:

   Analysis and Insights: Gloo Mesh Core recommendations for how to improve your Istio setups.

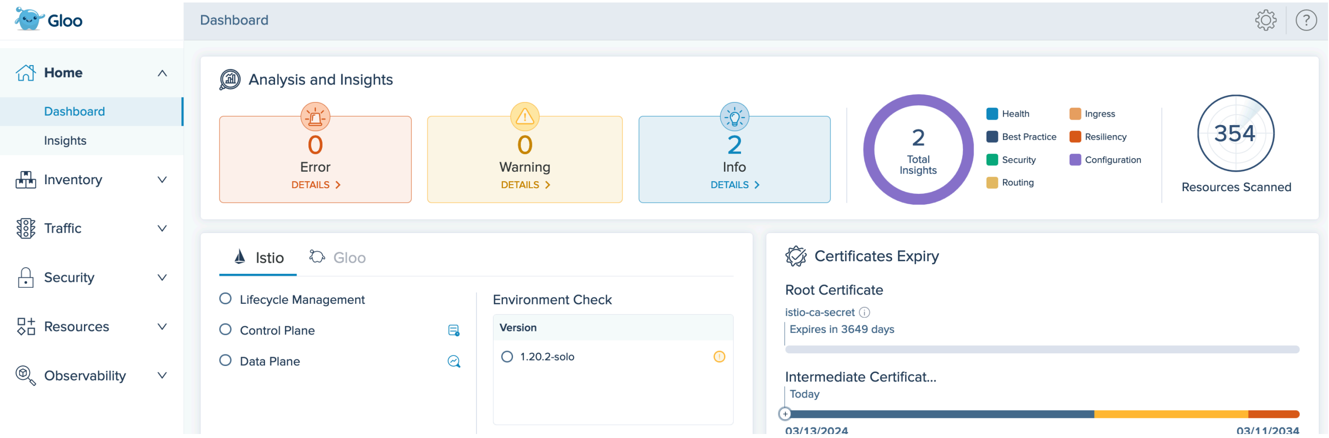
   Gloo and Istio health: A status check of the Gloo Mesh Core and Istio installations in each cluster.

   Certificates Expiry: Validity timelines for your root and intermediate Istio certificates.

   Cluster Services: Inventory of services across all clusters in your Gloo Mesh Core setup, and whether those services are in a service mesh or not.

   Istio FIPS: FIPS compliance checks for the `istiod` control planes and Istio data plane workloads.

   Zero Trust: Number of service mesh workloads that receive only mutual TLS (mTLS)-encrypted traffic, and number of external services that are accessed from the mesh.



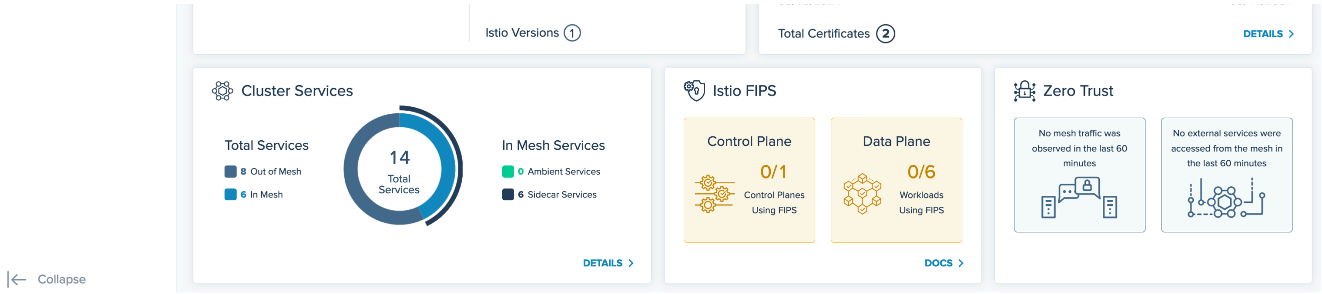


Figure: Gloo UI dashboard

Check insights

Review the insights for your environment. Gloo Mesh Core comes with an insights engine that automatically analyzes your Istio setups for health issues. These issues are displayed in the UI along with recommendations to harden your Istio setups. The insights give you a checklist to address issues that might otherwise be hard to detect across your environment.

1. On the **Analysis and Insights** card of the dashboard, you can quickly see a summary of the insights for your environment, including how many insights are available at each severity level, and the type of insight.

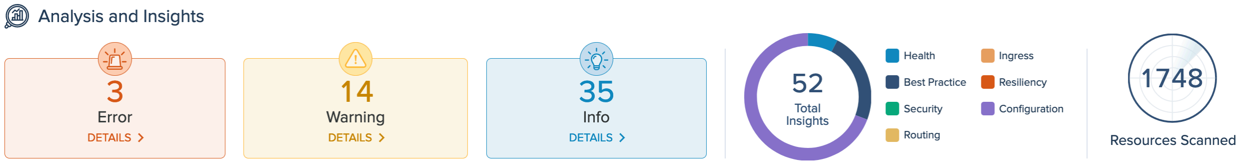
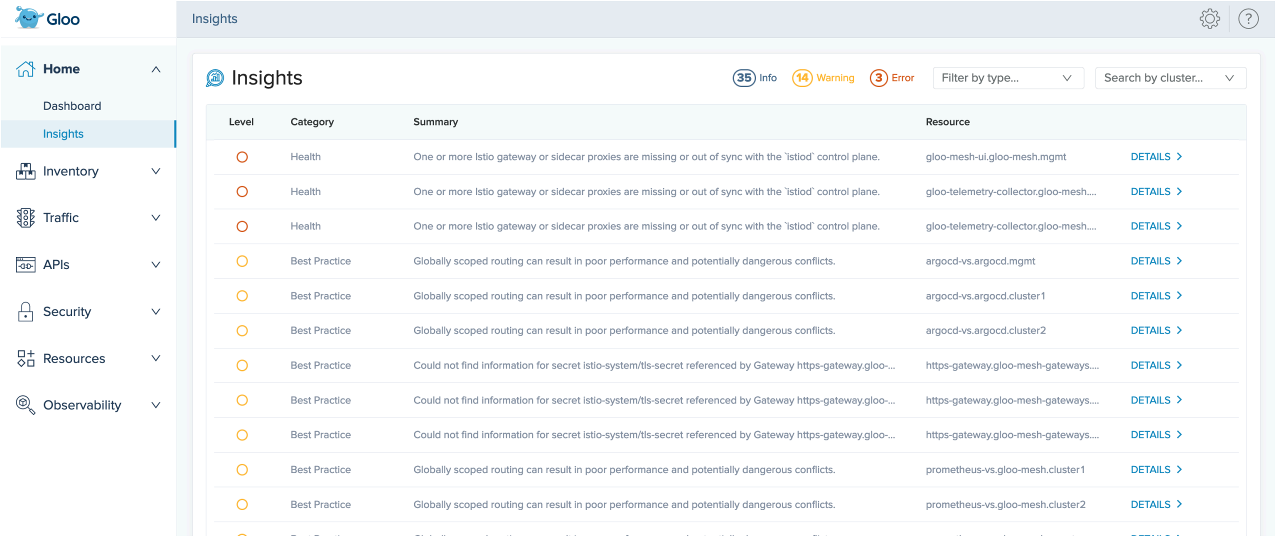


Figure: Insights and analysis card

2. View the list of insights by clicking the **Details** button, or go to the **Insights** page.

3. On the **Insights** page, you can view recommendations to harden your Istio setup, and steps to implement them in your environment. Gloo Mesh Core analyzes your setup, and returns individual insights that contain information about errors and warnings in your environment, best practices you can use to improve your configuration and security, and more.



Insights

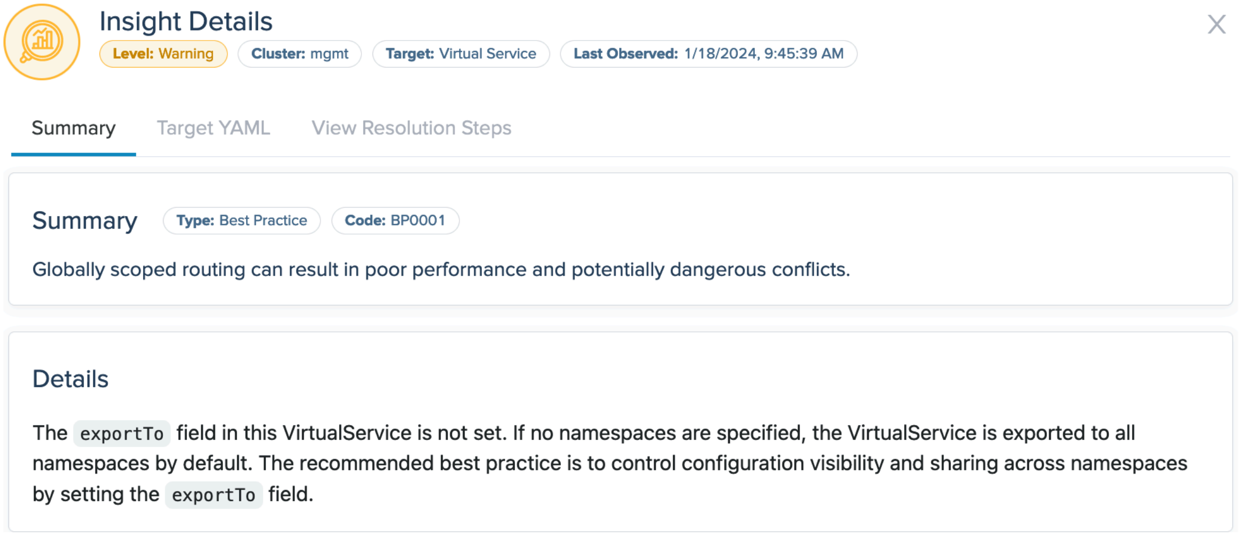
35 Info 14 Warning 3 Error Filter by type... Search by cluster...

Level	Category	Summary	Resource	
○	Health	One or more Istio gateway or sidecar proxies are missing or out of sync with the `istiod` control plane.	gloo-mesh-ui.gloo-mesh.mgmt	DETAILS >
○	Health	One or more Istio gateway or sidecar proxies are missing or out of sync with the `istiod` control plane.	gloo-telemetry-collector.gloo-mesh....	DETAILS >
○	Health	One or more Istio gateway or sidecar proxies are missing or out of sync with the `istiod` control plane.	gloo-telemetry-collector.gloo-mesh....	DETAILS >
○	Best Practice	Globally scoped routing can result in poor performance and potentially dangerous conflicts.	argocd-vs.argocd.mgmt	DETAILS >
○	Best Practice	Globally scoped routing can result in poor performance and potentially dangerous conflicts.	argocd-vs.argocd.cluster1	DETAILS >
○	Best Practice	Globally scoped routing can result in poor performance and potentially dangerous conflicts.	argocd-vs.argocd.cluster2	DETAILS >
○	Best Practice	Could not find information for secret istio-system/tls-secret referenced by Gateway https-gateway.gloo-...	https-gateway.gloo-mesh-gateways....	DETAILS >
○	Best Practice	Could not find information for secret istio-system/tls-secret referenced by Gateway https-gateway.gloo-...	https-gateway.gloo-mesh-gateways....	DETAILS >
○	Best Practice	Could not find information for secret istio-system/tls-secret referenced by Gateway https-gateway.gloo-...	https-gateway.gloo-mesh-gateways....	DETAILS >
○	Best Practice	Globally scoped routing can result in poor performance and potentially dangerous conflicts.	prometheus-vs.gloo-mesh.cluster1	DETAILS >
○	Best Practice	Globally scoped routing can result in poor performance and potentially dangerous conflicts.	prometheus-vs.gloo-mesh.cluster2	DETAILS >

Best Practice Globally scoped routing can result in poor performance and potentially dangerous conflicts. prometheus-vs.gloo-mesh.mgmt DETAILS >

Figure: Insights page

4 On an insight that you want to resolve, click **Details**. The details modal shows more data about the insight, such as the time when it was last observed in your environment, and if applicable, the extended settings or configuration that the insight applies to.



Insight Details

Level: Warning | Cluster: mgmt | Target: Virtual Service | Last Observed: 1/18/2024, 9:45:39 AM

Summary Type: Best Practice Code: BP0001

Globally scoped routing can result in poor performance and potentially dangerous conflicts.

Details

The `exportTo` field in this VirtualService is not set. If no namespaces are specified, the VirtualService is exported to all namespaces by default. The recommended best practice is to control configuration visibility and sharing across namespaces by setting the `exportTo` field.

Figure: Example insight

5 Click the **Target YAML** tab to see the resource file that the insight references, and click the **View Resolution Steps** tab to see guidance such as steps for fixing warnings and errors in your resource configuration or recommendations for improving your security and setup.

Next steps

Now that you have Gloo Mesh Core and Istio up and running, check out some of the following resources to learn more about Gloo Mesh Core and expand your service mesh capabilities.

Istio:

Find out more about hardened Istio `n-4` version support built into **Solo distributions of Istio**.

Check out the **Istio docs to configure and deploy Istio routing resources**.

Monitor and observe your Istio environment with Gloo Mesh Core's built-in **telemetry** tools.

When it's time to upgrade Istio, use Gloo Mesh Core to **upgrade managed Istio installations**.

For ambient installations, see **Upgrade ambient service meshes**.

Gloo Mesh Core:

Customize your Gloo Mesh Core installation with a **Helm-based setup**.

Explore **insights** to review and improve your setup's health and security posture.

When it's time to upgrade Gloo Mesh Core, see the **upgrade** guide.

Help and support:

- Talk to an expert to get advice or build out a proof of concept.
- Join the #gloo-mesh channel in the Solo.io community slack.
- Try out one of the Gloo workshops.

Cleanup

If you no longer need this quick-start Gloo Mesh Core environment, you can follow the steps in the uninstall guide.