


[Home](#) / [About](#) / [Home](#) / [Architecture](#) / Relay Architecture

Relay architecture

Review the Gloo relay server and agent architecture.

To support multicluster setups and provide exceptional multicluster routing, discovery, security, scalability, and observability capabilities across clusters, Gloo Mesh Gateway, Gloo Mesh Enterprise, and Gloo Mesh Core share the same relay architecture. The relay architecture consists of a Gloo management server and agents. The management server is commonly installed in a standalone management cluster. The agents are installed in each workload cluster. Workload clusters host your apps, and are registered with the management cluster.

 In testing or single-cluster environments, you can also run the Gloo management plane and data plane components in the same cluster.

Relay components

Gloo consists of a management server (sometimes called the “relay server”) for the management plane and relay agents for each workload cluster in the data plane.

Management server

After installation, a deployment named `gloo-mesh-mgmt-server` runs the management server. For your workload clusters to communicate with the management server, the `gloo-mesh-mgmt-server` service is automatically set up as a service of type `LoadBalancer` on a default port of `9900/TCP`. The server is responsible for configuring the Gloo agents in your workload cluster and maintaining the desired state of your environment. When you create Gloo custom resources, the server translates these to the appropriate open source custom resources that your Gloo product is based on, such as Istio or Envoy. Then, the server pushes config changes to the agents to apply in the workload clusters.

Management server replicas and clusters

By default, the management server is deployed with one replica. To increase availability, you can increase the number of replicas that you deploy in the management cluster.

Additionally, you can create multiple management clusters, and deploy one or more replicas of the management server to each cluster. For more information, see [High availability and disaster recovery](#).

Relay agents

After registration, a deployment named `gloo-mesh-agent` runs the relay agent on each workload cluster. The relay agent is exposed by the `gloo-mesh-agent` service on the default port `9977`. The agents send snapshots of the resources from each workload cluster to the management server.

Agent replicas

You can add replicas of the agent for higher availability. In such case, leader election affects which processes the agents handle. Logging and metric processes use the most resources, and scale as the number of services within the cluster grows.

The leader agent handles the following processes:

- Relay
- Discovery
- Certificate management
- Pod bouncing
- API discovery for Gloo Portal
- Schema reporter for Gloo Portal

All agent replicas, including the leader, handle the following processes:

- Access log sink
- Verifier cache clearing for CRD watch management

Relay communication

Communication between the management server and agents is initiated by the Gloo relay agents, which run in the workload clusters. The following figures outline the general flow of how the relay agents and server communicate to keep your configurations and environment up to date.

Note that these steps outline the relay process in a multicluster setup.

Workload cluster registration

The following image shows the process for registering a workload cluster with the management

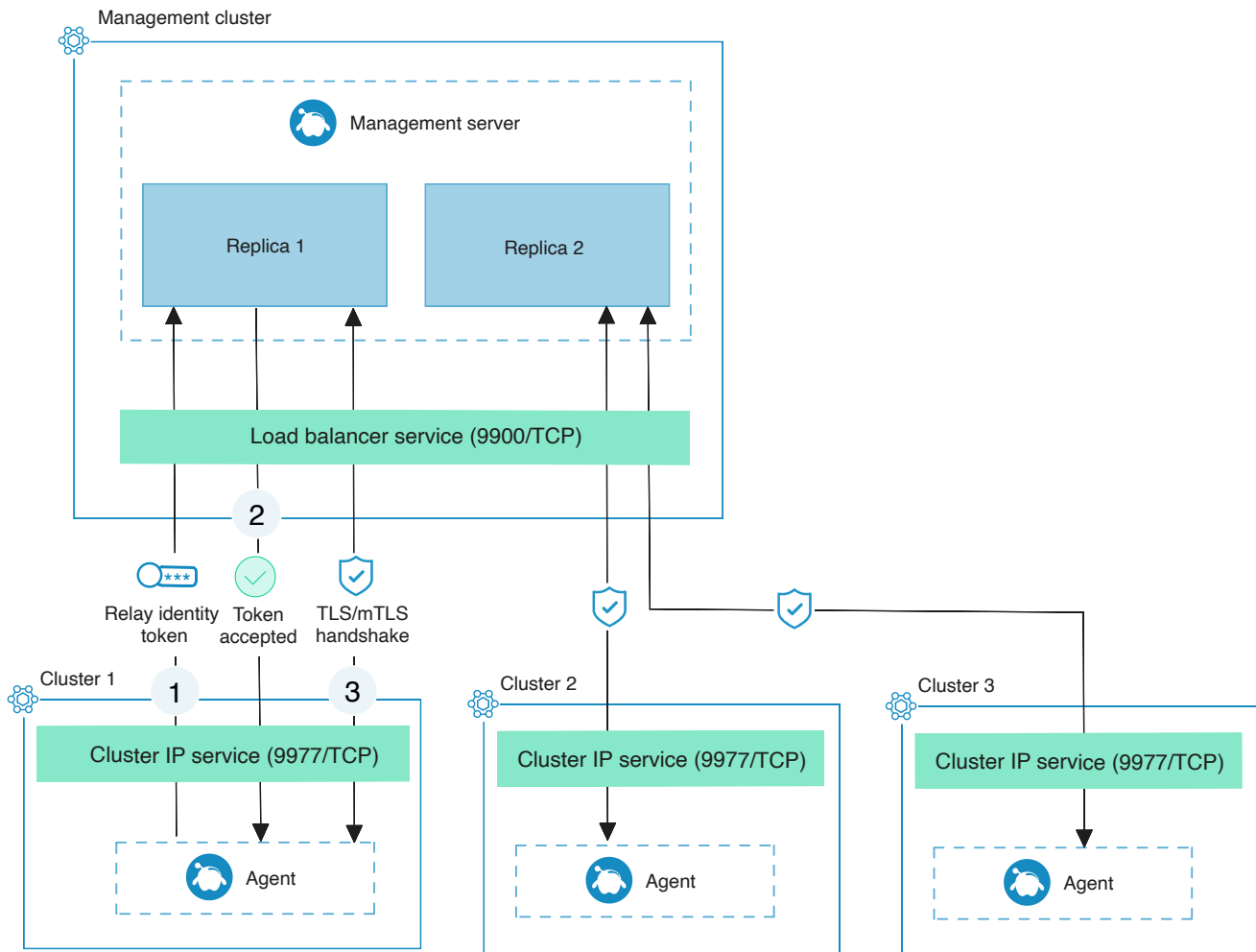


Figure: Registration of Gloo agents with the Gloo management server

- 1 During the workload cluster registration process, the Gloo agent in the workload cluster establishes a simple TLS connection to one of the Gloo management server replicas and sends a relay identity token. The relay identity token is a string value that is either generated during the installation of the Gloo management server and copied to the workload cluster, or provided by you in the form of a Kubernetes secret (mTLS setup) or environment variable (TLS setup) to the Gloo management server and agent.
- 2 The Gloo management server verifies the relay identity token value. If the agent's relay identity token matches the management server's token, the management server registers the agent. In mTLS setups, the management server also issues a client TLS certificate and sends it to the agent.
- 3 The Gloo agent initiates a TLS handshake. In mTLS setups, the Gloo agent verifies the identity of the Gloo management server and presents the client TLS certificate to prove its own identity. Only if both parties can be verified successfully, the TLS handshake is complete and the mTLS connection is established. In TLS setups, only the identity of the Gloo management server is verified.

After the connection is established, the Gloo agent gathers and sends its first discovery input snapshot. For more information about this process, see [Custom resource translation](#).

Management server and agent communication

The relay architecture provides several options for how you can secure the connection between the Gloo management server and agents. For testing environments, Gloo Mesh Enterprise generates self-signed TLS certificates that can be used to establish a simple or mutual TLS connection. For production environments, provide your own TLS certificates instead.


For an overview of supported options, see the [Setup options](#).

Gloo custom resource translation and reconciliation

Learn how Gloo custom resources are translated into Istio, Envoy, or Cilium resources and applied in each workload cluster.

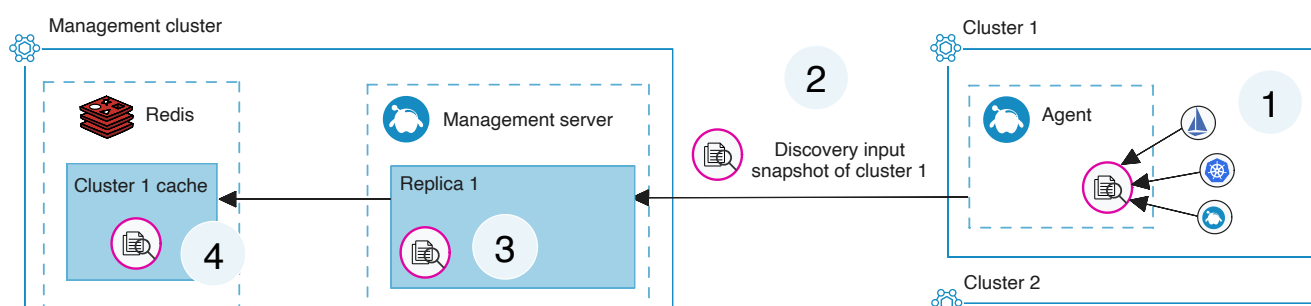
Translation process

The Gloo management server and agents use relay input and output snapshots to exchange information and reconcile the resources that must be applied in each cluster. The input snapshot includes all Gloo resources that the Gloo agent discovered on its local cluster as illustrated in the [Discovery input snapshot](#) tab. The Gloo management server translates these resources into Istio, Envoy, or Cilium resources and sends them as an output snapshot to the Gloo agent. To learn more about this process, see the [Translation and output snapshot](#) tab.

 To learn more about how translation works when you create Gloo custom resources in the management cluster vs. the workload clusters, see [Translation process with cluster context](#).

Discovery input snapshot

Translation and output snapshots



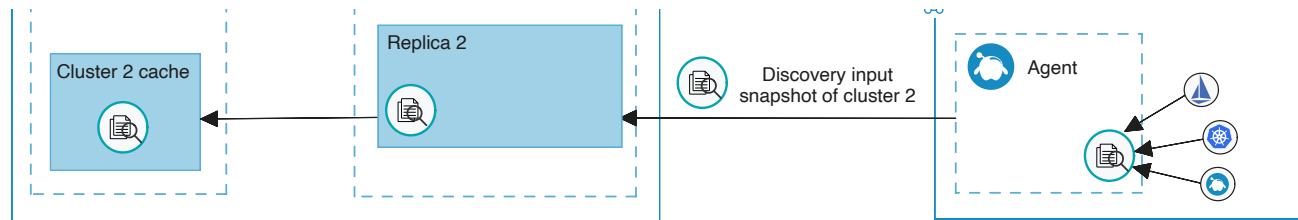


Figure: Discovery input snapshots are sent agents in each workload cluster to the Gloo management server

- 1 The relay agent in the workload cluster watches the resources that are created in the workload cluster. These resources include:

Discovered Kubernetes resources, such as Kubernetes services, deployments, replicaset, daemonsets, and statefulsets. The management server translates discovered resources into Istio resources and displays them in the Gloo UI. Note that you can use Istio discovery selectors to ignore certain Kubernetes resources. Ignored resources are not included in the snapshot that is sent from the agent to the management server.

Gloo custom resources that you create. The management server translates Gloo resources into Istio resources and displays them in the Gloo UI.

Istio resources, including:

Istio resources that, after initial server-agent setup, the management server automatically translates from your Gloo resources and writes back to the workload cluster. These resources are included in the snapshot to avoid accidentally deleting them from the workload cluster if an agent disconnects and reconnects, and to display them in the Gloo UI.

Any Istio resources that you manually created, so that they can be displayed in the Gloo UI.

Internal resources that are computed in memory by each agent and pushed to the management server without being persisted in the workload cluster. Internal resources include:

Gateway resources, which contain information about gateway endpoints in the cluster.

IssuedCertificate and **CertificateRequest** resources, which are used in internal multi-step workflows that involve both the agent and the management server.

- 2 The Gloo agent connects to one of the management server replicas and sends discovered resources as a discovery input snapshot to the management server. The input snapshot represents the current state of all discovered entities.
- 3 The Gloo management server saves the discovery input snapshot for all connected Gloo agents in the local memory.

- The Gloo management server saves the discovery input snapshot for their connected Gloo agents to Redis. Redis is a key component of the Gloo management plane. Then, the Kubernetes cluster resource is updated to reflect that the first snapshot for that workload cluster was received.

Translation process with cluster context

Review the following flow diagrams to learn more about how Gloo custom resources are translated when you apply Gloo resources in the management versus the workload clusters.

For more information about how Gloo translates custom resources, see [Custom resource translation](#).

Apply resources in the workload cluster

Apply resources in the management cluster

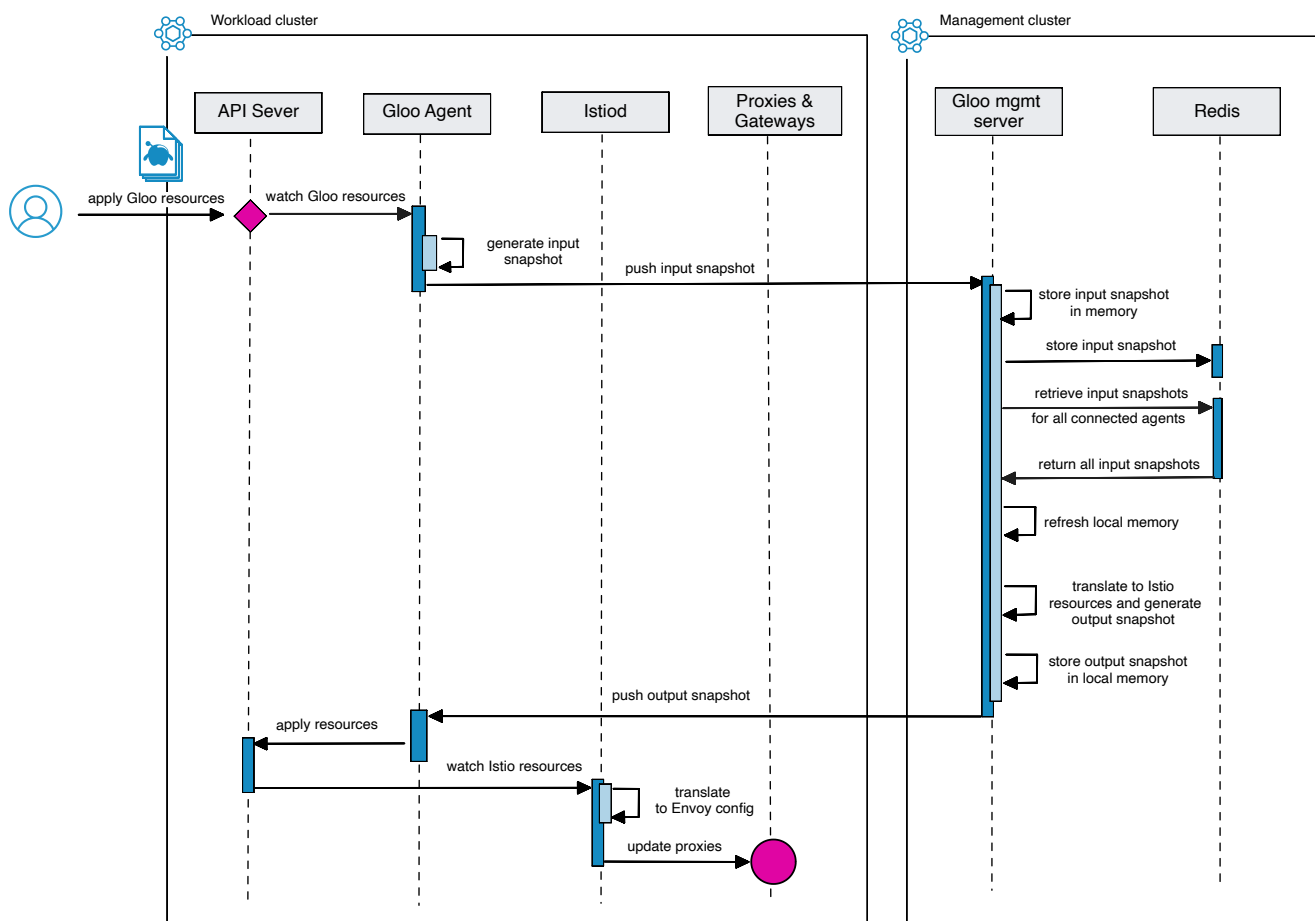


Figure: Translation process when resources are applied in the workload cluster

Translation failure scenarios

Translation can fail when individual components of the Gloo relay architecture become unavailable, such as during a restart. Review the following scenarios to understand the impact.

Redis becomes unavailable

Redis is a key component of the Gloo management plane and serves as the single source of truth for translating Gloo custom resources into Istio, Envoy, or Cilium resources. In the event that Redis becomes unavailable, translation continues as the Gloo management server uses the input snapshots that are stored in its local memory.

Flip through the following cards to learn more about this failure scenario.

Initial state

Redis unavailable

Redis back up

The following image shows the initial state. The Gloo management plane and control plane are in a healthy state and Gloo custom resource translation happens with the complete context of all workload clusters.

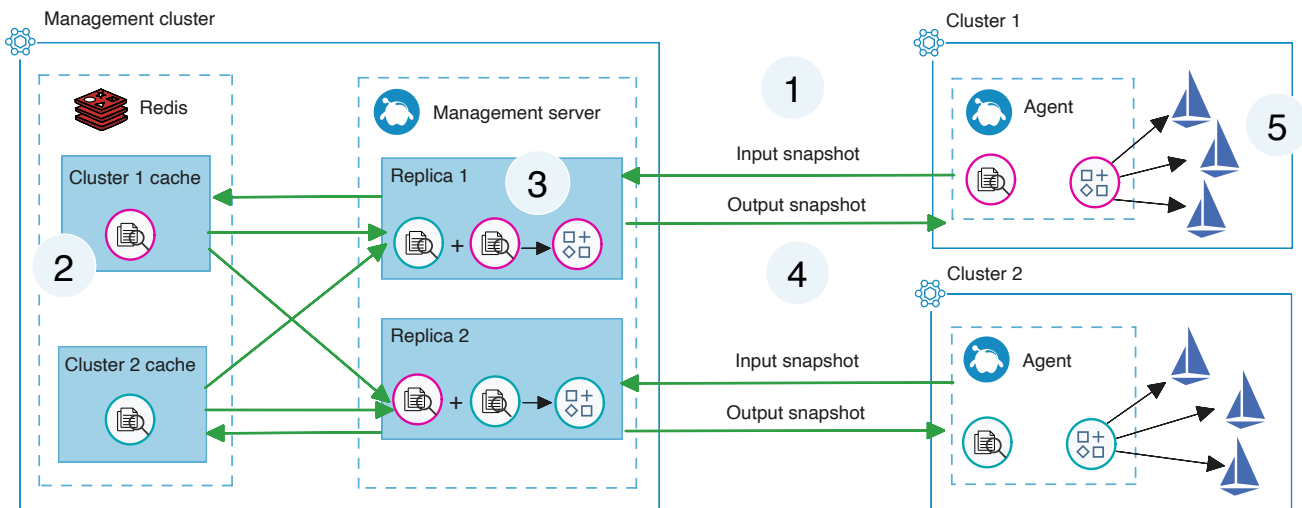


Figure: Initial state with a healthy Gloo management plane and data plane

- 1 The Gloo agents in each cluster gather all resources to create the initial discovery input snapshot. The Gloo agent connects to one of the management server replicas and sends discovered resources as a discovery input snapshot to the management server. The input snapshot represents the current state of all discovered entities.
- 2 The Gloo management server saves the discovery input snapshot for all connected Gloo agents in the local memory and in the Redis cache.
- 3 During the translation, the Gloo management server replica pulls all the discovery snapshots for all connected Gloo agents from Redis and stores them in the local memory. The Gloo management server then combines the input snapshots of all agents and decides what Istio, Envoy, or Cilium resources must be created on each workload cluster to fulfill the declared state. These resources are stored in a cluster-specific output snapshot.

- 4 The Gloo management server sends the output snapshot to each connected Gloo agent.
- 5 The Gloo agent reconciles the output snapshot against the Istio, Envoy, or Cilium resources that exist in the local workload cluster. The agent then creates, updates, and deletes Istio, Envoy, and Cilium resources to match the target configuration of the output snapshot.

Gloo agent disconnects

To translate Gloo custom resources based on the complete context of all workload clusters, each connected Gloo agent must send a discovery input snapshot with all discovered entities to the Gloo management server.

Flip through the cards to learn how translation continues if one of the Gloo agents disconnects.

Initial state

Gloo agent disconnects

Gloo agent reconnects

The following image shows the initial state. The Gloo management plane and control plane are in a healthy state and Gloo custom resource translation happens with the complete context of all workload clusters.

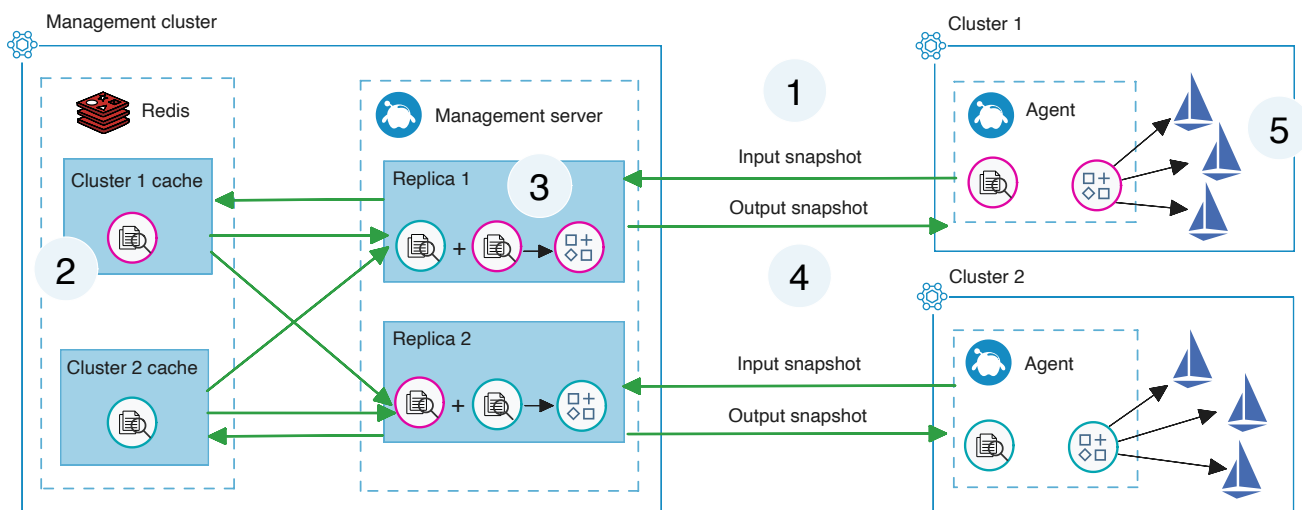


Figure: Initial state with a healthy Gloo management plane and data plane

- 1 The Gloo agents in each cluster gather all resources to create the initial discovery input snapshot. The Gloo agent connects to one of the management server replicas and sends discovered resources as a discovery input snapshot to the management server. The input snapshot represents the current state of all discovered entities.
- 2 The Gloo management server saves the discovery input snapshot for all connected Gloo agents in the local memory and in the Redis cache.
- 3 During the translation, the Gloo management server replica pulls all the discovery snapshots for all connected Gloo agents from Redis and stores them in the local memory. The Gloo

all connected Gloo agents from Redis and stores them in the local memory. The Gloo management server then combines the input snapshots of all agents and decides what Istio, Envoy, or Cilium resources must be created on each workload cluster to fulfill the declared state. These resources are stored in a cluster-specific output snapshot.

- 4 The Gloo management server sends the output snapshot to each connected Gloo agent.
- 5 The Gloo agent reconciles the output snapshot against the Istio, Envoy, or Cilium resources that exist in the local workload cluster. The agent then creates, updates, and deletes Istio, Envoy, and Cilium resources to match the target configuration of the output snapshot.

Gloo management server replica unavailable

When the Gloo agent connects to the Gloo management server, a management server replica is assigned and used to establish the simple TLS or mutual TLS connection with the Gloo agent. The Gloo agent then sends an input snapshot that contains all discovered entities from its local cluster.

Flip through the cards to learn how translation continues if the assigned Gloo management server replica becomes unavailable.

Initial state Gloo management server replica unavailable

Gloo management server replica back up

The following image shows the initial state. The Gloo management plane and control plane are in a healthy state and Gloo custom resource translation happens with the complete context of all workload clusters.

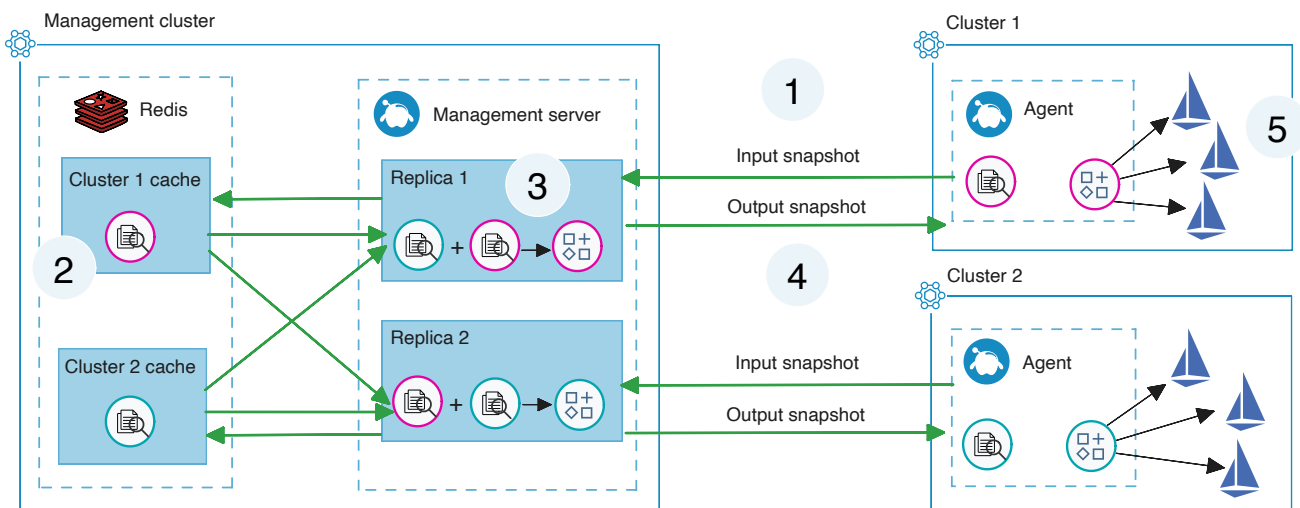


Figure: Initial state with a healthy Gloo management plane and data plane

- 1 The Gloo agents in each cluster gather all resources to create the initial discovery input snapshot. The Gloo agent connects to one of the management server replicas and sends

discovered resources as a discovery input snapshot to the management server. The input snapshot represents the current state of all discovered entities.

- 2 The Gloo management server saves the discovery input snapshot for all connected Gloo agents in the local memory and in the Redis cache.
- 3 During the translation, the Gloo management server replica pulls all the discovery snapshots for all connected Gloo agents from Redis and stores them in the local memory. The Gloo management server then combines the input snapshots of all agents and decides what Istio, Envoy, or Cilium resources must be created on each workload cluster to fulfill the declared state. These resources are stored in a cluster-specific output snapshot.
- 4 The Gloo management server sends the output snapshot to each connected Gloo agent.
- 5 The Gloo agent reconciles the output snapshot against the Istio, Envoy, or Cilium resources that exist in the local workload cluster. The agent then creates, updates, and deletes Istio, Envoy, and Cilium resources to match the target configuration of the output snapshot.

Redis and Gloo management server restart

In the following scenario, Redis and the Gloo management server become unavailable at the same time, which removes all input snapshots from the Redis cache and the management server's local memory.



In versions 2.5.3, 2.4.1 and earlier, partial translation could occur if Gloo agents reconnected at different times. Starting in 2.6.0, **safe mode** is automatically enabled on the Gloo management server that prevents incomplete output snapshots from being sent to connected workload clusters.

Flip through the cards to learn more about this failure scenario.

Initial state

Redis unavailable

Management server unavailable

Redis and mgmt server back up, one agent connects

The following image shows the initial state. The Gloo management plane and control plane are in a healthy state and Gloo custom resource translation happens with the complete context of all workload clusters.



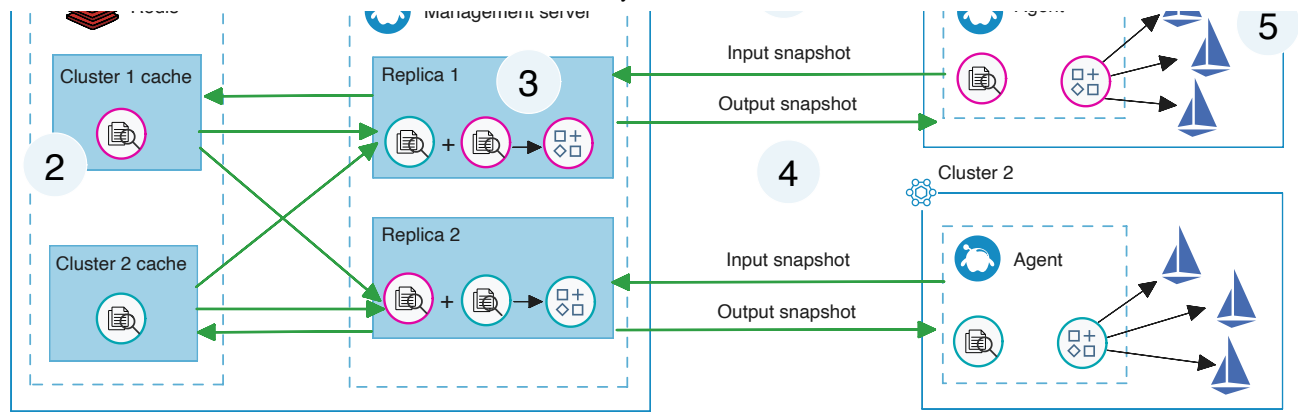


Figure: Initial state with a healthy Gloo management plane and data plane

- 1 The Gloo agents in each cluster gather all resources to create the initial discovery input snapshot. The Gloo agent connects to one of the management server replicas and sends discovered resources as a discovery input snapshot to the management server. The input snapshot represents the current state of all discovered entities.
- 2 The Gloo management server saves the discovery input snapshot for all connected Gloo agents in the local memory and in the Redis cache.
- 3 During the translation, the Gloo management server replica pulls all the discovery snapshots for all connected Gloo agents from Redis and stores them in the local memory. The Gloo management server then combines the input snapshots of all agents and decides what Istio, Envoy, or Cilium resources must be created on each workload cluster to fulfill the declared state. These resources are stored in a cluster-specific output snapshot.
- 4 The Gloo management server sends the output snapshot to each connected Gloo agent.
- 5 The Gloo agent reconciles the output snapshot against the Istio, Envoy, or Cilium resources that exist in the local workload cluster. The agent then creates, updates, and deletes Istio, Envoy, and Cilium resources to match the target configuration of the output snapshot.

Safe mode

Starting in version 2.6.0, safe mode is enabled on the Gloo management server by default to ensure that the server translates input snapshots only if all input snapshots are present in Redis or its local memory. This way, translation only occurs based on a complete translation context.

In version 2.5.3, 2.4.11, and earlier a race condition was identified that could trigger during simultaneous restarts of the management plane and Redis, including an upgrade to a newer Gloo Mesh Enterprise version. If hit, this failure mode can lead to partial translations on the Gloo management server which can result in Istio resources being temporarily deleted from the output snapshots that are sent to the Gloo agents. To prevent this issue in later releases, safe mode is now

enabled by default in version 2.6.0 and later.

Safe mode can be configured in two ways:

Safe mode

Safe start window



Safe mode is applied to workload clusters that were initially connected to the Gloo management server and sent at least one input snapshot. If a workload cluster is registered, but the Gloo agent never connected to the Gloo management server to send its first input snapshot, the cluster is not included in safe mode. You can check whether or not a workload cluster successfully sent an input snapshot by reviewing the `KubernetesCluster` resource that represents the workload cluster. For more information, see [KubernetesCluster resource](#).

Option 1: Safe mode

In the event that Redis and the Gloo management server restart simultaneously and only some agents reconnect successfully to re-populate their input snapshots in Redis and the management server's local memory, the management server halts translation. Translation does not resume until the agents in each workload cluster reconnect to the management server and their input snapshots are re-populated in Redis. Until translation resumes, the agents use the last provided output snapshot.

To enable this setting, follow these general upgrade steps. Note that safe mode is enabled by default in version 2.6.0 and later.

- 1 Scale down the number of Gloo management server pods to 0.

```
kubectl scale deployment gloo-mesh-mgmt-server --replicas=0 -n
```

- 2 **Upgrade your Gloo Mesh Enterprise installation.** Add the following settings in the Helm values file for the Gloo management plane.

```
glooMgmtServer:  
  safeMode: true
```

- 3 Scale the Gloo management server back up to the number of desired replicas. The following example uses 1 replica.

```
kubectl scale deployment gloo-mesh-mgmt-server --replicas=1 -n
```

Review how safe mode works in the event that Redis and the Gloo management server restart, and only some of the Gloo agents reconnect. For more information about this scenario, see [Redis and Gloo management server restart](#).

Without safe mode (disabled by default in 2.5.x and earlier)

With safe mode (enabled by default in 2.6.0 and later)

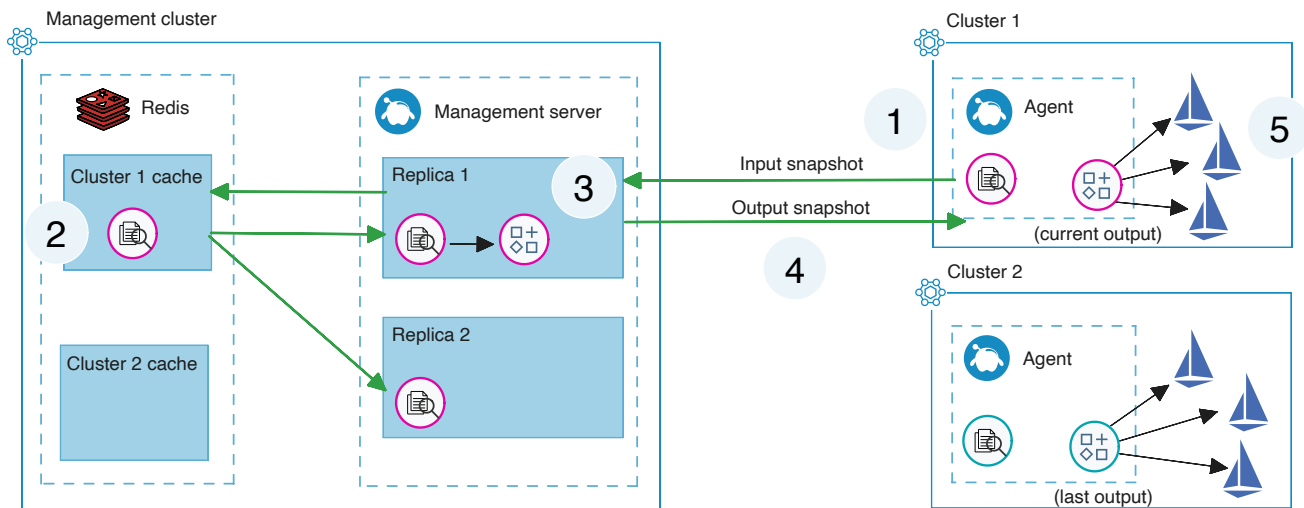


Figure: Redis and the Gloo management server come back up healthy, only one Gloo agent reconnects. Translation continues

- 1 Redis and the Gloo management server recover and come back up healthy. The Gloo agent in cluster 1 reconnects successfully to the Gloo management server and sends its input snapshot. However, the Gloo agent in cluster 2 takes longer to reconnect.
- 2 The input snapshot of cluster 1 is stored in the local memory on the Gloo management server and in Redis.
- 3 During the translation, both management server replicas pull all input snapshots from Redis. Because cluster 2 is still not connected to the management server, only the input snapshot of cluster 1 is available in Redis and can be pulled. The management server replica 1 starts the translation of the input snapshot and generates an output snapshot without the resource context of cluster 2.
- 4 The Gloo management server sends the output snapshot to the Gloo agent in cluster 1. No output snapshot is sent to cluster 2, because cluster 2 is still disconnected.
- 5 The Gloo agents in cluster 1 and cluster 2 continue to reconcile the output snapshot against the local Envoy resources that exist in the local workload cluster. Because the output snapshot was generated without the resource context of cluster 2, the agents in cluster 2 continue to reconcile against their local state.

Istio or Envoy resources that exist in the local workload cluster. Because the output snapshot was built without the context of cluster 2, the agent might remove important Istio resources that are required for cross-cluster communication.

Option 2: Safe start window

With safe mode, the Gloo management server halts translation until the input snapshots of all workload clusters that were previously connected to the Gloo management server are present in the Redis cache. However, if clusters have connectivity issues, translation might be halted for a long time, even for healthy clusters. You might want translation to resume after a certain period of time, even if some input snapshots are missing. To do so, use the `glooMgmtServer.safeStartWindow` field in your Gloo management server Helm values file.

The safe start window represents the time in seconds that the Gloo management server halts translation until the Gloo agents of all workload clusters connect and send their input snapshots to populate the Redis cache. As such, the safe start window behaves the same as safe mode. After the time expires, the management server starts translation by using the input snapshots that are currently present in Redis. Missing snapshots are not included in the output snapshot for each workload cluster.



To set a safe start window, you must disable safe mode.

To set a safe start window, follow these general upgrade steps. Note that you must disable safe mode for the safe start window to take effect.

- 1 Scale down the number of Gloo management server pods to 0.

```
kubectl scale deployment gloo-mesh-mgmt-server --replicas=0 -n
```

- 2 **Upgrade your Gloo Mesh Enterprise installation.** Add the following settings in the Helm values file for the Gloo management plane.

```
glooMgmtServer:
  safeMode: false
  safeStartWindow: 90
```

- 3 Scale the Gloo management server back up to the number of desired replicas. The following example uses 1 replica.

```
kubectl scale deployment gloo-mesh-mgmt-server --replicas=1 -n
```

If you do not want the management server to wait, you can set this option to 0 (zero). However, keep in mind that setting this option to 0 can lead to incomplete output snapshots in multicluster setups if safe mode is disabled at the same time.

Exclude clusters from safe mode

You can optionally exclude clusters from safe mode by adding the `skipWarming` option to the `KubernetesCluster` custom resource that represents the cluster that you want to exclude. The `skipWarming` setting instructs the Gloo management server to not trigger safe mode and to start the translation, even if the Redis cache was not populated with the latest input snapshot for that cluster.

```
kubectl apply --context $MGMT_CONTEXT -f- <<EOF
apiVersion: admin.gloo.solo.io/v2
kind: KubernetesCluster
metadata:
  name: $REMOTE_CLUSTER1
  namespace: gloo-mesh
  labels:
    env: prod
spec:
  clusterDomain: cluster.local
  skipWarming: true
EOF
```

For example, you might want to register a new workload cluster. During the registration process, the `KubernetesCluster` is created. However, the agent in the workload cluster cannot connect to the management server due to a connectivity issue. If safe mode is turned on, the translation for all workload clusters halts until the connectivity issue is resolved and an input snapshot is sent from the newly registered cluster so that the Redis cache is populated. You can prevent this scenario by setting `skipWarming: true` in the `KubernetesCluster` resource of the workload cluster that you want to register. After the workload cluster connects successfully, you can remove this setting or explicitly set `skipWarming: false` to include the cluster in the safe mode.



Clusters that never connected to the Gloo management server and sent at least one input snapshot are automatically excluded from safe mode. For more information, see

[KubernetesCluster resource](#).

Indefinite safe mode

Safe mode is intended to be a temporary state that is triggered when multiple management plane components are simultaneously restarted. Most of the time, safe mode resolves automatically as soon as all the management plane components come back healthy and the Redis cache is re-populated with the input snapshots of all agents.

However, safe mode might not get automatically resolved due to the following reasons:

- One or more Gloo agents fail to reconnect with the Gloo management server, such as due to failing to schedule, insufficient resources, network errors, or getting rejected by the management server due to capacity issues.

- Redis remains unstable, such as due to insufficient resources.

- A KubernetesCluster resource was added to the Gloo management server. However, the corresponding Gloo agent is not yet set up on the workload cluster.

If any of these conditions are met, safe mode might stay intact, and translation is halted indefinitely for all connected Gloo agents. To resolve this issue, you can choose between the following options:

- Resolve the issue that prevents the Gloo agent from reconnecting. For example, if the cluster that the Gloo agent is deployed to has insufficient resources, add the required resources. If the error is due to issue in the network, try to resolve the network error.

- Deregister the affected workload cluster by removing the KubernetesCluster resource that represents the workload cluster.

- Exclude the affected cluster from safe mode.**

Monitor safe mode

You have multiple options to monitor if safe mode was triggered in your cluster.

Gloo UI

If a workload cluster triggered the Gloo management server to run in safe mode, a banner is shown in the Gloo UI. The banner includes the details about the workload clusters that triggered safe mode. You can use this information to investigate connectivity issues for these clusters and to troubleshoot potential Redis or Gloo management server issues.

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: For more information, see the [CLI documentation](#).

```
meshctl dashboard
```

kubect:

- 1 Port-forward the `gloo-mesh-ui` service on 8090.

```
kubect port-forward -n gloo-mesh svc/gloo-mesh-ui 8090:809
```

- 2 Open your browser and connect to <http://localhost:8090>.

Metrics

You can use the `gloo_mesh_safe_mode_active` metric to determine if your environment currently runs in safe mode and the workload clusters that triggered safe mode. This metric is automatically collected by the Gloo telemetry pipeline and you can use the built-in Prometheus server to view this metric or run queries on that metric.

- 1 Port-forward the Prometheus pod in your cluster to open the Prometheus expression browser.

meshctl **kubect**

```
meshctl proxy prometheus
```

- 2 Run the following query to determine if your environment is currently running in safe mode, and the workload cluster that triggered it.

```
count by(cluster) (sum by(cluster) (gloo_mesh_safe_mode_active
```

- 3 When you run multiple replicas of the Gloo management server, you might have some replicas in safe mode and others are not. You can use the following Prometheus query to determine which workload clusters are currently connected to a management replica that runs in safe mode.

```
count by(cluster) ((relay_pull_clients_connected != 0) * on(po
```

KubernetesCluster resource

Safe mode is applied to workload clusters that were initially connected to the Gloo management server and sent at least one input snapshot. If a workload cluster is registered, but the Gloo agent never connected to the Gloo management server to send its first input snapshot, the cluster is not included in safe mode.

To find out if a cluster was connected to the Gloo management server, you can review the condition of the KubernetesCluster resource that represents the workload cluster. The KubernetesCluster resource can have the following conditions:

Reason	Status	Type	Description	Included in safe mode?
ClusterRegistered	False	ClusterWarm	The KubernetesCluster resource is created, but no snapshot was received from the Gloo agent yet.	✗
FirstSnapshotReceived	True	ClusterWarm	The Gloo agent connected successfully to the Gloo management server and sent its first input snapshot. Note that this status is final and does not change, even if the Gloo agent disconnects or the Gloo management server becomes unavailable.	✓

The following snippet shows an example for a cluster that successfully sent an input snapshot and therefore is included in safe mode.

```
conditions:
- lastTransitionTime: "2024-04-30T15:47:06.011421879Z"
  message: The agent successfully connected to the management s
    input snapshot was stored in Redis
  reason: FirstSnapshotReceived
  status: "True"
  type: ClusterWarm
```

