Inbound Forwarding

Shared with Istio Community

Owner: howardjohn **Working Group:** Networking	**Status**: WIP | In Review | **Approved** | Obsolete **Created**: 01/27/2021 **Approvers**: Yuchen [x], samnaser [x], Costin [x]

TL;DR

Rework our inbound clusters to allow seamless adoption of Istio to a wider range of applications, while avoiding unintentionally exposing users applications.

Background

When an application is deployed in Kubernetes (or elsewhere), it must bind to some address. Common binds include:

- Wildcard bind (0.0.0.0, ::)
- Localhost bind (127.0.0.1, ::1)
- Pod IP bind (1.2.3.4, for example. Typically pod IP is derived from downward API)

Istio's inbound cluster configuration is configured to send requests to ports exposed through a Service to 127.0.0.1:port, while requests to unknown ports follow the InboundPassthroughCluster path, configured as an ORIGINAL_DST cluster.

This leads to the following differences between a standard Kubernetes cluster and an Istio-enabled cluster for Pod <-> Pod communication:

Red: requests fail. Green: requests succeeds

No Istio (Service)	**No Istio (Non-Service)**	**Istio (Service)**	**Istio (Non-service)**
Wildcard bind	Wildcard bind	Wildcard bind	Wildcard bind
Localhost bind	Localhost bind	Localhost bind	Localhost bind
Pod IP bind	Pod IP bind	Pod IP bind	Pod IP bind

The disconnect in the "Istio (Service)" case leads to unexpected behavior when adopting Istio

- Applications binding to pod IP will immediately break with no clear indication about what is wrong. Most issues in List of applications incompatible with Istio, including Zookeeper and ElasticSearch, can be attributed to this.
- Applications binding to localhost may do so for security reasons. When adopting Istio, they will now be exposed to other pods, which may be undesirable. Note: this would require the user to add it to a Service, which doesn't really make much sense, so any case of this happening is likely accidental.

Requirements

- Reachability behavior should be the same with and without Istio out of the box
    - This implies all columns of the table above are identical, and applications like Zookeeper work out of the box
- Performance should be in line with the existing implementation
- Users should have the flexibility to retain existing behavior, if desired

Design

XDS Changes

Inbound clusters will be changed from STATIC to ORIGINAL_DST. Additionally, the UpstreamBindConfig will be modified to the "magic" 127.0.0.6 address. This is to avoid iptables loops; we have some special logic that will allow calls from Envoy to go back to themselves so that for calls from app -> app, we hit outbound and inbound paths. The 127.0.0.6 address allows short-circuiting this by returning early if the source IP is 127.0.0.6.

This new cluster configuration will match the existing inbound passthrough cluster exactly.

WIP implementation

Readiness Probes

Currently, readiness probes suffer a similar issue. While Kubernetes will typically send these requests to the pod IP, the istio-agent will intercept these and forward to localhost, exhibiting the same behavior as real traffic.

These can be changed to unconditionally send to the Pod IP (which we know from injecting into an environment variable via downward API and used in various other locations today.

One issue with this is the probing logic doesn't have access to the Sidecar resource to be customizable by `Sidecar.Ingress` like the inbound clusters. However, this is already the behavior today (for example, we don't support a UDS readiness probe). Since this change is

100% compatible with existing Kubernetes behavior, which is the goal of the app probe rewrite, this is not considered an issue.

TODO: look into getting local address in go, use that

User Impact

For new users, this change should be strictly an improvement; more of their existing applications will work out of the box when they install Istio.

The primary concern here is backwards incompatibility with existing users. Generally, all users will be binding to wildcard or pod IP. However, users may end up binding to localhost because:

1. They only tested with Istio enabled, and it happened to work, so they stuck with it.
2. Our documentation recommended explicitly binding to localhost; this has since been updated.

Because of this, care will need to be taken around upgrades. See upgrade section below.

For all users, the source IP of proxied requests will now show as 127.0.0.6 instead of 127.0.0.1. This is unlikely to impact users, unless they have some special logic to detect if they are running behind Envoy that checks the source IP is "127.0.0.1" or similar, which is expected to be unlikely.

API

The `Sidecar.Ingress` API currently allows controlling the binding configuration. For example:

```
ingress:
- port:
    number: 8080
    protocol: HTTP
    name: http
  defaultEndpoint: 127.0.0.1:8080
```

Other supported options for `defaultEndpoint` are: `127.0.0.1:PORT`, `0.0.0.0:PORT` (which will forward to the instance IP), or `unix:///path/to/socket`.

We can extend the API to allow omitting the `defaultEndpoint` option, which would give the standard ORIGINAL_DST behavior to allow specifying existing `Sidecar.Ingress` settings without changing the cluster bind configuration.

Users that *want* the current behavior of sending to localhost can continue to do so by configuring `defaultEndpoint: 127.0.0.1`. This is often desired for VMs.

Note: the `0.0.0.0:PORT` (which will forward to the instance IP) syntax was introduced in 1.9 to solve this same problem. If desired, it's not too late to change or remove it.

Upgrade

The main challenge with this design is ensuring we do not break existing users when migrating.

In order to detect localhost binds, a script will be provided: check-binds.sh.
This script will query all pods in the mesh to determine the bind type for all ports exposed by a service. This is done by querying `istioctl` for ports, then running `kubectl exec -- ss` inside the proxy container and parsing the result.
For example:

```
Checking cassandra-1611606400-0/default...
  Port 7000: other bind found (10.36.2.100)
  Port 7001: bind not found
  Port 7199: localhost bind
  Port 8080: bind not found
  Port 9042: wildcard bind
  Port 9160: wildcard bind
```

Running this tool will enable Istio operators to understand if this change will negatively impact them.

Timeline

- Istio 1.9: no action
- Istio 1.10:
    - write a blog post explaining the future change with a migration guide. This will instruct users to run the script and/or manually identify pods that need changes, and change them to wildcard binds or use the Sidecar API. These changes can be safely made in old Istio versions, so they will continue to work before and after upgrade. This ensures users can migrate their applications without requiring atomically changing them with the Istio version.
    - enable the changes to the Sidecar api documented above
    - The cluster changes will be merged but disabled by default (via environment variable). This will allow users to test out the change early.
- Istio 1.11:
    - Change is enabled by default
    - Upgrade notice with the same content as the blog (including migration guide)
- Istio 1.12+:
    - Consider removing the environment variable.

User Feedback

To attempt to gauge the impact this would have no existing users, I asked a few large users the impact this would have.

- 1 large user ran the script and found a few localhost binds, which are assumed to be unintentional. However, their use of `Sidecar.Ingress` today means there would be no change anyways.
- 2 large users ran the script and found no issues.
- 2 large users did not run the script, but indicated the change would not be a concern. Both desire localhost binds for VMs and are already using `Sidecar.Ingress` today to meet that concern.
- Please add/comment/reach out to me if you have any other feedback

Test Plan

- Ensure all existing integration tests, other than the existing negative test case.
- Extend `echo` app to allow localhost binds, and test all variants of `Sidecar.Ingress` x `application bind` x `Service Exposure`.
- Measure the performance of the change in cluster type using the standard Istio proxy benchmarks.

Performance

Tests were performed with passthrough and static cluster. Latency, throughput, memory, and CPU were analyzed. There was no significant difference between the two modes. As a result, this change is considered to have "passed" the performance test and is not expected to have any impact (positive or negative).