**5 Things to Know About Istio, the Open Source Service Mesh**

<https://thenewstack.io/5-things-to-know-about-istio-the-open-source-service-mesh/>

## 1. Automatic Sidecar Injection

Istio requires Kubernetes v1.9 or newer because of the [Custom Resource](https://kubernetes.io/docs/concepts/api-extension/custom-resources/) and [mutating webhooks](https://kubernetes.io/docs/admin/admission-controllers/#mutatingadmissionwebhook-beta-in-19) support. If your cluster has the mutating webhook feature enabled, I recommend you install the Istio automatic sidecar injector, which automatically injects the [Envoy](https://www.envoyproxy.io/) sidecar for all microservice deployments you want Istio to manage for you. [IBM Cloud Container Service](https://www.ibm.com/cloud/container-service?cm_mmc=OSocial_Blog-_-Cloud_Cloud+Platform-_-WW_WW-_-Istio&cm_mmca1=000023UA&cm_mmca2=10007995&) clusters (v1.9 or newer) have the mutating webhooks feature enabled by default, which works nicely with the Istio automatic sidecar injector. As this is still a new feature in the overall scheme of Istio development, I recommend that you use it with caution.

## 2. Traffic Management Improvement

In Istio v0.1, you could use ingress route rules to specify which kinds of traffic you want coming in through your microservice. Now you can also use egress route rules to specify which kinds of traffic can go out of the service and which external service it can communicate with.  For example, you could easily configure your service to talk to an IBM Watson service in IBM Cloud to empower your service with machine learning capabilities.  I recently modernized the Kubernetes guestbook example to automatically generate an emoji based on machine learning and learned that traffic management really helped me test and roll out the new version. I highly recommend you explore traffic management in your development and staging environments so you can be confident in your testing.

## 3. Improved Telemetry

As an Istio user, I don’t have to do anything to enable various metrics or tracing, which gives me valuable time back. I can deploy my microservice application and let Istio manage it without having to change any part of my application or my deployment .yaml file. The Zipkin or [Jaeger](https://github.com/jaegertracing/jaeger) tracing tools provide detailed tracing information for each of the requests that come into my application and allow me to drill into each request to view how traffic flows end-to-end within the service mesh.  If you want your trace spans tied together for each given request, you’ll need to update your code to propagate a few [headers](https://istio.io/docs/tasks/telemetry/distributed-tracing.html#understanding-what-happened). I highly recommend you consider telemetry as the first Istio feature to roll into your production environment.

## 4. Multi-Environment Support (Consule, Eureka and VM)

One of the initial goals for Istio was to enable multi-environment support because you can’t require all workloads to run in Kubernetes in a microservice architecture. Some of your applications may run in Kubernetes, while some may run in Docker Swarm or VMs. Since last October, Istio has advanced to provide early support for VMs, integration with some of the more popular service discovery systems such as Consul and Eureka, and has expanded to support other runtime environments. The community is also working on multi-cloud and multicluster support. I’m very excited to see Istio expanded to multiple vendor clouds.

## 5. Mutual TLS (mTLS) and Its Flexible Configuration

This one is my personal favorites! Istio enhances the microservices to communicate securely via mTLS, without the need for any code changes to your microservices. What I find most interesting is that the community made this flexible and easy to use by introducing the per service mTLS enablement or disablement, so you can adjust this configuration on a per-service basis. Further, any user can plug in any existing CA certificate and key or configure role-based access control (RBAC) for each service. As much as I love these security features, I recommend adopting Istio without mTLS enabled first, then gradually enable mTLS. It can be much harder to troubleshoot problems when mTLS is permitted so you want to make sure your microservices can work with Istio without mTLS enabled first.